Package 'lefko3'

May 8, 2026

Type Package

Title Historical and Ahistorical Population Projection Matrix Analysis

Version 6.7.3

Date 2026-04-24

Description Complete analytical environment for the construction and analysis of matrix population models and integral projection models. Includes the ability to construct historical matrices, which are 2d matrices comprising 3 consecutive times of demographic information. Estimates both raw and function-based forms of historical and standard ahistorical matrices. It also estimates function-based age-by-stage matrices and raw and function-based Leslie matrices.

Encoding UTF-8

License GPL (>= 2)

URL https://github.com/dormancy1/lefko3

Imports Rcpp (>= 1.0.5), glmmTMB, lme4, MASS, Matrix, methods, MuMIn, pscl, rlang, stats, VGAM, grDevices

LinkingTo Rcpp, RcppArmadillo, BH

LazyData true

BugReports https://github.com/dormancy1/lefko3/issues

RoxygenNote 7.3.3

Suggests knitr, popbio, rmarkdown, Rcompadre

VignetteBuilder knitr

NeedsCompilation yes

Author Richard P. Shefferson [aut, cre] (ORCID: <https://orcid.org/0000-0002-5234-3131>), Johan Ehrlen [aut] (ORCID: <https://orcid.org/0000-0001-8539-8967>)

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

Depends R (>= 3.5.0)

Repository CRAN

Date/Publication 2026-04-24 09:40:02 UTC

Contents

`lefko3-package` *Historical and Ahistorical Population Projection Matrix Analysis*

Description

This package creates population matrix projection models (MPMs) for use in population ecological analyses. It presents a complete working environment for the construction and analysis of ALL kinds of MPMs and IPMs, including age, stage, and age-by-stage versions. Its specialty is the estimation of historical MPMs, which are 2d matrices comprising 3 monitoring occasions (2 time steps or periods) of demographic information. The package constructs both function-based and raw MPMs for both standard ahistorical (i.e. 2 occasions, 1 time step) and historical analyses, has functions for complex density-dependent and independent, and stochastic and cyclical, projections, and also includes the automatic calculation of quality control metrics throughout every step of analysis. It also includes powerful functions to standardize demographic datasets.

Details

The lefko3 package provides seven categories of functions:

1. Data transformation and handling functions
2. Functions determining population characteristics from vertical data
3. Model building and selection
4. Matrix / integral projection model creation functions
5. Population dynamics analysis and projection functions
6. Functions describing, summarizing, or visualizing MPMs and derived structures
7. Extra functions used to illustrate core theory and ideas.

lefko3 also includes example datasets complete with sample code.

Author(s)

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

Authors:

- Johan Ehrlen (ORCID)

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

Johan Ehrlén

References

Shefferson, R.P., J. Ehrlen, and S. Kurokawa. 2021. *lefko3*: analyzing individual history through size-classified matrix population models. *Methods in Ecology and Evolution* 12(2): 378-382.

See Also

Useful links:

- https://github.com/dormancy1/lefko3
- Report bugs at https://github.com/dormancy1/lefko3/issues

`actualstage3` *Calculate Actual Stage, Age, Stage-Pair, or Age-Stage Distributions*

Description

Function `actualstage3()` shows the frequencies and proportions of each stage, stage pair, age-stage, or age in each year.

Usage

```
actualstage3(
  data,
  check_stage = TRUE,
  check_age = FALSE,
  historical = FALSE,
  year2 = NULL,
  indices = NULL,
  stagecol = NULL,
  agecol = NULL,
  remove_stage = NULL,
  t1_allow = NULL
)
```

Arguments

`data`	A demographic dataset in hfv format.
`check_stage`	A logical value indicating whether to assess frequencies and proportions of stages. Defaults to `TRUE`.
`check_age`	A logical value indicating whether to assess frequencies and proportions of ages. Defaults to `FALSE`.
`historical`	A logical value indicating whether the stage structure should be ahistorical (`FALSE`) or historical (`TRUE`). Defaults to `FALSE`.
`year2`	A string value indicating the name of the variable coding for monitoring occasion at time *t*. Defaults to `"year2"`.

indices	A vector of three strings, indicating the stage indices for times *t*+1, *t*, and *t*-1, respectively, in data. Defaults to c("stage3index", "stage2index", "stage1index").
stagecol	A vector of three strings, indicating the stage name columns for times *t*+1, *t*, and *t*-1, respectively, in data. Defaults to stagecol = c("stage3", "stage2", "stage1").
agecol	A single string indicating the age of individuals in time *t*. Defaults to "obsage".
remove_stage	A string vector indicating the names of stages to remove from consideration. Defaults to "NotAlive".
t1_allow	A string vector indicating which stages to be removed should be allowed in the stage at time *t*-1 portion of historical stage pairs, if historical = TRUE. Defaults to "NotAlive". Can also be set to "none".

Value

A data frame with the following variables:

rowid	A string identifier term, equal to the monitoring occasion in time *t* and the stage index.
stageindex	The stageframe index of the stage. Only output if check_stage = TRUE.
stage	The name of each stage, or NA. Only output if check_stage = TRUE.
stage2	The name of the stage in time *t*. Only output if check_stage = TRUE.
stage1	The name of the stage in time *t*-1, or NA. Only output if check_stage = TRUE.
age	The age at time *t*. Only output if check_age = TRUE.
year2	Monitoring occasion in time *t*.
frequency	The number of individuals in the respective stage and time.
actual_prop	The proportion of individuals alive in time *t* in the respective stage.

Notes

This function produces frequencies and proportions of stages in hfv formatted data using stage index variables rather than stage name variables, and so requires the former. The latter is only required if the user wants to know the associated stage names.

Frequencies and proportions will be calculated for all times, including the last time, which is generally found in the stage3 columns of the last year2 entry in object data. The default is to treat the year2 entry for that time as max(year2) + 1.

If check_stage = TRUE and check_age = FALSE, then this function will assess frequencies and proportions of stages or historical stage-pairs. If both check_stage = TRUE and check_age = TRUE, then this function will assess frequencies and proportions of age-stages. If check_stage = FALSE and check_age = TRUE, then the frequencies and proportions of ages only will be assessed.

Note that no stageframe is required for this function to operate. Stage names and their order are inferred directly from the object data.

Examples

```
sizevector <- c(0, 0, 0, 0, 0, 0, 1, 3, 6, 11, 19.5)
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, 1.5, 1.5, 3.5, 5)
comments <- c("Dormant seed", "1st yr protocorm", "2nd yr protocorm",
  "3rd yr protocorm", "Seedling", "Dormant adult",
  "Extra small adult (1 shoot)", "Small adult (2-4 shoots)",
  "Medium adult (5-7 shoots)", "Large adult (8-14 shoots)",
  "Extra large adult (>14 shoots)")
cypframe_raw <- sf_create(sizes = sizevector, stagenames = stagevector,
  repstatus = repvector, obsstatus = obsvector, matstatus = matvector,
  propstatus = propvector, immstatus = immvector, indataset = indataset,
  binhalfwidth = binvec, comments = comments)

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, age_offset = 4)

all_stage_props_ah <- actualstage3(cypraw_v1)
all_stage_props_h <- actualstage3(cypraw_v1, historical = TRUE)
all_stage_props_h_NANotAllow <- actualstage3(cypraw_v1, historical = TRUE,
  t1_allow = "none")
all_stage_props_as <- actualstage3(cypraw_v1, check_age = TRUE)
all_age_props <- actualstage3(cypraw_v1, check_stage = FALSE,
  check_age = TRUE)
```

---

add_lM *Add Matrices to a lefkoMat or lefkoMatList Object*

Description

Function add_lM() adds matrices to lefkoMat and lefkoMatList objects.

Usage

```
add_lM(
  lM,
  Amats = NA,
```

```
  Umats = NA,
  Fmats = NA,
  UFdecomp = FALSE,
  entrystage = 1,
  pop = NA,
  patch = NA,
  year = NA
)
```

Arguments

lM	The lefkoMat or lefkoMatList object to add matrices to.
Amats	Either a single A matrix, or a list of A matrices. Not necessary if Umats and Fmats are both provided.
Umats	Either a single U matrix, or a list of U matrices. Not necessary if Amats and Fmats are both provided, or if UFdecomp = TRUE and entrystage is provided.
Fmats	Either a single F matrix, or a list of U matrices. Not necessary if Amats and Umats are both provided, or if UFdecomp = TRUE and entrystage is provided.
UFdecomp	A logical value indicating whether U and F matrices should be inferred from A matrices and the given entrystage. Defaults to TRUE.
entrystage	The stage or stages produced by reproductive individuals. Used to determine which transitions are reproductive for U-F decomposition. Defaults to 1, which corresponds to the first stage in the stageframe.
pop	The population designation for each matrix. If object lM includes only a single population, then defaults to that designation. Otherwise requires a designation as input.
patch	The patch designation for each matrix. If object lM includes only a single patch, then defaults to that designation. Otherwise requires a designation as input.
year	The designation for occasion at time *t* corresponding to each matrix. Cannot be left empty.

Value

A lefkoMat or lefkoMatList object incorporating the new matrices within the object input in lM. Note that if a lefkoMatList object is used as input, then ALL composite lefkoMat objects will have the same matrices added in exactly the same way.

Notes

This function will not allow matrices of different dimension from those input in object lM to be added to that object.

Two of Amats, Umats, and Fmats must be provided for this function to proceed. Also, if Amats, Umats, and Fmats are all provided, then this function will default to replacing Amats with the sum of the respective Umats and Fmats.

See Also

create_lM()

delete_lM()

subset_lM()

Examples

```
# These matrices are of 9 populations of the plant species Anthyllis
# vulneraria, and were originally published in Davison et al. (2010) Journal
# of Ecology 98:255-267 (doi: 10.1111/j.1365-2745.2009.01611.x).

sizevector <- c(1, 1, 2, 3) # These sizes are not from the original paper
stagevector <- c("Sdl", "Veg", "SmFlo", "LFlo")
repvector <- c(0, 0, 1, 1)
obsvector <- c(1, 1, 1, 1)
matvector <- c(0, 1, 1, 1)
immvector <- c(1, 0, 0, 0)
propvector <- c(0, 0, 0, 0)
indataset <- c(1, 1, 1, 1)
binvec <- c(0.5, 0.5, 0.5, 0.5)

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

# POPN C 2003-2004
XC3 <- matrix(c(0, 0, 1.74, 1.74,
0.208333333, 0, 0, 0.057142857,
0.041666667, 0.076923077, 0, 0,
0.083333333, 0.076923077, 0.066666667, 0.028571429), 4, 4, byrow = TRUE)

# 2004-2005
XC4 <- matrix(c(0, 0, 0.3, 0.6,
0.32183908, 0.142857143, 0, 0,
0.16091954, 0.285714286, 0, 0,
0.252873563, 0.285714286, 0.5, 0.6), 4, 4, byrow = TRUE)

# 2005-2006
XC5 <- matrix(c(0, 0, 0.50625, 0.675,
0, 0, 0, 0.035714286,
0.1, 0.068965517, 0.0625, 0.107142857,
0.3, 0.137931034, 0, 0.071428571), 4, 4, byrow = TRUE)

# POPN E 2003-2004
XE3 <- matrix(c(0, 0, 2.44, 6.569230769,
0.196428571, 0, 0, 0,
0.125, 0.5, 0, 0,
0.160714286, 0.5, 0.133333333, 0.076923077), 4, 4, byrow = TRUE)

XE4 <- matrix(c(0, 0, 0.45, 0.646153846,
```

```
0.06557377, 0.090909091, 0.125, 0,
0.032786885, 0, 0.125, 0.076923077,
0.049180328, 0, 0.125, 0.230769231), 4, 4, byrow = TRUE)

XE5 <- matrix(c(0, 0, 2.85, 3.99,
0.083333333, 0, 0, 0,
0, 0, 0, 0,
0.416666667, 0.1, 0, 0.1), 4, 4, byrow = TRUE)

mats_list <- list(XC3, XC4, XC5, XE3, XE4, XE5)
yr_ord <- c(1, 2, 3, 1, 2, 3)
pch_ord <- c(1, 1, 1, 2, 2, 2)

anth_lefkoMat <- create_lM(mats_list, anthframe, hstages = NA,
  historical = FALSE, poporder = 1, patchorder = pch_ord, yearorder = yr_ord)

XH3 <- matrix(c(0, 0, 0.1125, 1.05,
0.2, 0, 0, 0,
0, 0.5, 0, 0,
0.2, 0.5, 0, 0), 4, 4, byrow = TRUE)

XH3u <- matrix(c(0, 0, 0, 0,
0.2, 0, 0, 0,
0, 0.5, 0, 0,
0.2, 0.5, 0, 0), 4, 4, byrow = TRUE)

XH4 <- matrix(c(0, 0, 0, 0,
0, 0, 0.5, 0,
0.8, 0.5, 0.25, 0.25,
0.2, 0, 0, 0.75), 4, 4, byrow = TRUE)

XH4u <- matrix(c(0, 0, 0, 0,
0, 0, 0.5, 0,
0.8, 0.5, 0.25, 0.25,
0.2, 0, 0, 0.75), 4, 4, byrow = TRUE)

XH5 <- matrix(c(0, 0, 0.2, 1.05,
0, 0, 0, 0,
0.001, 0.001, 0.333333333, 0,
0.001, 0, 0, 0), 4, 4, byrow = TRUE)

XH5u <- matrix(c(0, 0, 0, 0,
0, 0, 0, 0,
0.001, 0.001, 0.333333333, 0,
0.001, 0, 0, 0), 4, 4, byrow = TRUE)

anth_lefkoMat <- add_lM(anth_lefkoMat, Amats = list(XH3, XH4, XH5),
  Umats = list(XH3u, XH4u, XH5u), patch = c(3, 3, 3), year = c(1, 2, 3))
```

---

add_stage *Add a New Stage to an Existing lefkoMat or lefkoMatList Object*

Description

Function add_stage() adds a new stage to an existing lefkoMat or lefkoMatList object. In addition to altering the ahstages object within the MPM, it alters the hstages and agestages objects and adds the appropriate number of new rows and columns depending on the kind of MPM input. Note that, if entering a lefkoMatList object, then a stage will be added to all lefkoMat objects contained therein.

Usage

```
add_stage(mpm, add_before = 0L, add_after = 0L, stage_name = NULL)
```

Arguments

mpm	The lefkoMat or lefkoMatList object to add a stage to.
add_before	The index of the stage to insert a new stage before. This index should be derived from the ahstages of the input mpm. Cannot be set if add_after is to be used.
add_after	The index of the stage to insert a new stage after. This index should be derived from the ahstages of the input mpm. Cannot be set if add_before is to be used.
stage_name	The name of the new stage to add. Defaults to new_stage.

Value

A new copy of the original MPM edited to include new rows and columns in the associated matrices, and with ahstages, agestages, and hstages objects edited to include the new stage.

See Also

edit_lM()

Examples

```
data(cypdata)

cyp_lesl_data <- 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",
  stagesize = "sizeadded", NAas0 = TRUE, age_offset = 2)

cyp_survival <- glm(alive3 ~ obsage + as.factor(year2), data = cyp_lesl_data,
  family = "binomial")
cyp_fecundity <- glm(feca2 ~ 1 + obsage + as.factor(year2),
  data = cyp_lesl_data, family = "poisson")

mod_params <- create_pm(name_terms = TRUE)
mod_params$modelparams[22] <- "obsage"

germination <- 0.08
protocorm_to_seedling <- 0.10
seeding_to_adult <- 0.20
```

```
seeds_per_fruit <- 8000

cyp_lesl_supp <- supplemental(historical = FALSE, stagebased = FALSE,
  agebased = TRUE, age2 = c(1, 2), type = c(1, 1),
  givenrate = c(protocorm_to_seedling, seeding_to_adult))

cyp_lesl_fb_mpm <- fleslie(data = cyp_lesl_data, surv_model = cyp_survival,
  fec_model = cyp_fecundity, paramnames = mod_params, last_age = 7,
  fecage_min = 3, fecmod = (germination * seeds_per_fruit),
  supplement = cyp_lesl_supp)

altered1 <- add_stage(cyp_lesl_fb_mpm, add_before = 1, stage_name = "DS")
```

`aflefko2` *Create Function-based Ahistorical Age x Stage Matrix Projection Model*

Description

Function `aflefko2()` returns ahistorical age x stage MPMs corresponding to the patches and occasions given, including the associated component transition and fecundity matrices, data frames detailing the characteristics of ahistorical stages and the exact age-stage combinations corresponding to rows and columns in estimated matrices, and a data frame characterizing the patch and occasion combinations corresponding to these matrices.

Usage

```
aflefko2(
  year = "all",
  patch = "all",
  stageframe,
  supplement = NULL,
  repmatrix = NULL,
  overwrite = NULL,
  data = NULL,
  modelsuite = NULL,
  surv_model = NULL,
  obs_model = NULL,
  size_model = NULL,
  sizeb_model = NULL,
  sizec_model = NULL,
  repst_model = NULL,
  fec_model = NULL,
  jsurv_model = NULL,
  jobs_model = NULL,
  jsize_model = NULL,
  jsizeb_model = NULL,
```

```
  jsizec_model = NULL,
  jrepst_model = NULL,
  jmatst_model = NULL,
  paramnames = NULL,
  inda = NULL,
  indb = NULL,
  indc = NULL,
  annua = NULL,
  annub = NULL,
  annuc = NULL,
  surv_dev = 0,
  obs_dev = 0,
  size_dev = 0,
  sizeb_dev = 0,
  sizec_dev = 0,
  repst_dev = 0,
  fec_dev = 0,
  jsurv_dev = 0,
  jobs_dev = 0,
  jsize_dev = 0,
  jsizeb_dev = 0,
  jsizec_dev = 0,
  jrepst_dev = 0,
  jmatst_dev = 0,
  density = NA,
  fecmod = 1,
  random.inda = FALSE,
  random.indb = FALSE,
  random.indc = FALSE,
  final_age = NA,
  continue = TRUE,
  prebreeding = TRUE,
  negfec = FALSE,
  ipm_method = "CDF",
  reduce = FALSE,
  simple = FALSE,
  err_check = FALSE,
  exp_tol = 700,
  theta_tol = 1e+08,
  sparse_output = FALSE
)
```

Arguments

year	A variable corresponding to the observation occasion, or a set of such values, given in values associated with the year term used in linear model development. Defaults to "all", in which case matrices will be estimated for all occasions.
patch	A variable designating which patches or subpopulations will have matrices esti-

mated. Defaults to "all", but can also be set to specific patch names or a vector thereof.

stageframe — An object of class stageframe. These objects are generated by function sf_create(), and include information on the size, observation status, propagule status, reproduction status, immaturity status, maturity status, stage group, size bin widths, and other key characteristics of each ahistorical stage.

supplement — An optional data frame of class lefkoSD that provides supplemental data that should be incorporated into the MPM. Three kinds of data may be integrated this way: transitions to be estimated via the use of proxy transitions, transition overwrites from the literature or supplemental studies, and transition multipliers for survival and fecundity. This data frame should be produced using the supplemental() function. Can be used in place of or in addition to an overwrite table (see overwrite below) and a reproduction matrix (see repmatrix below).

repmatrix — An optional reproduction matrix. This matrix is composed mostly of 0s, with non-zero entries acting as element identifiers and multipliers for fecundity (with 1 equaling full fecundity). If left blank, and no supplement is provided, then aflefko2() will assume that all stages marked as reproductive produce offspring at 1x that of estimated fecundity, and that offspring production will yield the first stage noted as propagule or immature. Must be the dimensions of an ahistorical stage-based matrix.

overwrite — An optional data frame developed with the overwrite() function describing transitions to be overwritten either with given values or with other estimated transitions. Note that this function supplements overwrite data provided in supplement.

data — The historical vertical demographic data frame used to estimate vital rates (class hfvdata), which is required to initialize times and patches properly. Variable names should correspond to the naming conventions in verticalize3() and historicalize3(). Not required if option modelsuite is set to a vrm_input object.

modelsuite — One of three kinds of lists. The first is a lefkoMod object holding the vital rate models and associated metadata. The second is a lefkoModList object, which is a list of lefkoMod objects generally created to conduct a bootstrapped MPM analysis. Alternatively, an object of class vrm_input may be provided. If given, then surv_model, obs_model, size_model, sizeb_model, sizec_model, repst_model, fec_model, jsurv_model, jobs_model, jsize_model, jsizeb_model, jsizec_model, jrepst_model, jmatst_model, and paramnames are not required. One or more of these models should include size or reproductive status in occasion *t*-1. Although this is optional input, it is recommended, and without it all vital rate model inputs (named XX_model) are required.

surv_model — A linear model predicting survival probability. This can be a model of class glm or glmer, and requires a predicted binomial variable under a logit link. Ignored if modelsuite is provided. This model must have been developed in a modeling exercise testing only the impacts of occasion *t*.

obs_model — A linear model predicting sprouting or observation probability. This can be a model of class glm or glmer, and requires a predicted binomial variable under

a logit link. Ignored if `modelsuite` is provided. This model must have been developed in a modeling exercise testing only the impacts of occasion *t*.

`size_model` A linear model predicting primary size. This can be a model of class `glm`, `glmer`, `glmmTMB`, `zeroinfl`, `vglm`, `lm`, or `lmer`. Ignored if `modelsuite` is provided. This model must have been developed in a modeling exercise testing only the impacts of occasion *t*.

`sizeb_model` A linear model predicting secondary size. This can be a model of class `glm`, `glmer`, `glmmTMB`, `zeroinfl`, `vglm`, `lm`, or `lmer`. Ignored if `modelsuite` is provided. This model must have been developed in a modeling exercise testing only the impacts of occasion *t*.

`sizec_model` A linear model predicting tertiary size. This can be a model of class `glm`, `glmer`, `glmmTMB`, `zeroinfl`, `vglm`, `lm`, or `lmer`. Ignored if `modelsuite` is provided. This model must have been developed in a modeling exercise testing only the impacts of occasion *t*.

`repst_model` A linear model predicting reproduction probability. This can be a model of class `glm` or `glmer`, and requires a predicted binomial variable under a logit link. Ignored if `modelsuite` is provided. This model must have been developed in a modeling exercise testing only the impacts of occasion *t*.

`fec_model` A linear model predicting fecundity. This can be a model of class `glm`, `glmer`, `glmmTMB`, `zeroinfl`, `vglm`, `lm`, or `lmer`. Ignored if `modelsuite` is provided. This model must have been developed in a modeling exercise testing only the impacts of occasion *t*.

`jsurv_model` A linear model predicting juvenile survival probability. This can be a model of class `glm` or `glmer`, and requires a predicted binomial variable under a logit link. Ignored if `modelsuite` is provided. This model must have been developed in a modeling exercise testing only the impacts of occasion *t*.

`jobs_model` A linear model predicting juvenile sprouting or observation probability. This can be a model of class `glm` or `glmer`, and requires a predicted binomial variable under a logit link. Ignored if `modelsuite` is provided. This model must have been developed in a modeling exercise testing only the impacts of occasion *t*.

`jsize_model` A linear model predicting juvenile primary size. This can be a model of class `glm`, `glmer`, `glmmTMB`, `zeroinfl`, `vglm`, `lm`, or `lmer`. Ignored if `modelsuite` is provided. This model must have been developed in a modeling exercise testing only the impacts of occasion *t*.

`jsizeb_model` A linear model predicting juvenile secondary size. This can be a model of class `glm`, `glmer`, `glmmTMB`, `zeroinfl`, `vglm`, `lm`, or `lmer`. Ignored if `modelsuite` is provided. This model must have been developed in a modeling exercise testing only the impacts of occasion *t*.

`jsizec_model` A linear model predicting juvenile tertiary size. This can be a model of class `glm`, `glmer`, `glmmTMB`, `zeroinfl`, `vglm`, `lm`, or `lmer`. Ignored if `modelsuite` is provided. This model must have been developed in a modeling exercise testing only the impacts of occasion *t*.

`jrepst_model` A linear model predicting reproduction probability of a mature individual that was immature in time *t*. This can be a model of class `glm` or `glmer`, and requires

a predicted binomial variable under a logit link. Ignored if `modelsuite` is provided. This model must have been developed in a modeling exercise testing only the impacts of occasion *t*.

`jmatst_model` A linear model predicting maturity probability of an individual that was immature in time *t*. This can be a model of class `glm` or `glmer`, and requires a predicted binomial variable under a logit link. Ignored if `modelsuite` is provided. This model must have been developed in a modeling exercise testing only the impacts of occasion *t*.

`paramnames` A data frame with three columns, the first describing all terms used in linear modeling, the second (must be called `mainparams`) giving the general model terms that will be used in matrix creation, and the third showing the equivalent terms used in modeling (must be named `modelparams`). Function `create_pm()` can be used to create a skeleton `paramnames` object, which can then be edited. Only required if `modelsuite` is not supplied.

`inda` Can be a single value to use for individual covariate `a` in all matrices, a pair of values to use for times *t* and *t*-1 in historical matrices, or a vector of such values corresponding to each occasion in the dataset. Defaults to `NULL`.

`indb` Can be a single value to use for individual covariate `b` in all matrices, a pair of values to use for times *t* and *t*-1 in historical matrices, or a vector of such values corresponding to each occasion in the dataset. Defaults to `NULL`.

`indc` Can be a single value to use for individual covariate `c` in all matrices, a pair of values to use for times *t* and *t*-1 in historical matrices, or a vector of such values corresponding to each occasion in the dataset. Defaults to `NULL`.

`annua` Can be a single value to use for annual covariate `a` in all matrices, a pair of values to use for times *t* and *t*-1 in historical matrices, or a vector of such values corresponding to each occasion in the dataset. Defaults to `NULL`.

`annub` Can be a single value to use for annual covariate `b` in all matrices, a pair of values to use for times *t* and *t*-1 in historical matrices, or a vector of such values corresponding to each occasion in the dataset. Defaults to `NULL`.

`annuc` Can be a single value to use for annual covariate `c` in all matrices, a pair of values to use for times *t* and *t*-1 in historical matrices, or a vector of such values corresponding to each occasion in the dataset. Defaults to `NULL`.

`surv_dev` A numeric value to be added to the y-intercept in the linear model for survival probability. Defaults to `0`.

`obs_dev` A numeric value to be added to the y-intercept in the linear model for observation probability. Defaults to `0`.

`size_dev` A numeric value to be added to the y-intercept in the linear model for primary size. Defaults to `0`.

`sizeb_dev` A numeric value to be added to the y-intercept in the linear model for secondary size. Defaults to `0`.

`sizec_dev` A numeric value to be added to the y-intercept in the linear model for tertiary size. Defaults to `0`.

`repst_dev` A numeric value to be added to the y-intercept in the linear model for probability of reproduction. Defaults to `0`.

fec_dev	A numeric value to be added to the y-intercept in the linear model for fecundity. Defaults to 0.
jsurv_dev	A numeric value to be added to the y-intercept in the linear model for juvenile survival probability. Defaults to 0.
jobs_dev	A numeric value to be added to the y-intercept in the linear model for juvenile observation probability. Defaults to 0.
jsize_dev	A numeric value to be added to the y-intercept in the linear model for juvenile primary size. Defaults to 0.
jsizeb_dev	A numeric value to be added to the y-intercept in the linear model for juvenile secondary size. Defaults to 0.
jsizec_dev	A numeric value to be added to the y-intercept in the linear model for juvenile tertiary size. Defaults to 0.
jrepst_dev	A numeric value to be added to the y-intercept in the linear model for juvenile reproduction probability. Defaults to 0.
jmatst_dev	A numeric value to be added to the y-intercept in the linear model for juvenile maturity probability. Defaults to 0.
density	A numeric value indicating density value to use to propagate matrices. Only needed if density is an explanatory term used in one or more vital rate models. Defaults to NA.
fecmod	A scalar multiplier of fecundity. Defaults to 1.0.
random.inda	A logical value denoting whether to treat individual covariate a as a random, categorical variable. Otherwise is treated as a fixed, numeric variable. Defaults to FALSE.
random.indb	A logical value denoting whether to treat individual covariate b as a random, categorical variable. Otherwise is treated as a fixed, numeric variable. Defaults to FALSE.
random.indc	A logical value denoting whether to treat individual covariate c as a random, categorical variable. Otherwise is treated as a fixed, numeric variable. Defaults to FALSE.
final_age	The final age to model in the matrix, where the first age will be age 0 if post-breeding, and 1 if pre-breeding. Defaults to the maximum age in the dataset.
continue	A logical value designating whether to allow continued survival of individuals past the final age noted in the stageframe, using the demographic characteristics of the final age. Defaults to TRUE.
prebreeding	A logical value indicating whether the life history model is a pre-breeding model. Defaults to TRUE.
negfec	A logical value denoting whether fecundity values estimated to be negative should be reset to 0. Defaults to FALSE.
ipm_method	A string indicating what method to use to estimate size transition probabilities, if size is treated as continuous. Options include: "midpoint", which utilizes the midpoint method; and "CDF", which uses the cumulative distribution function. Defaults to "CDF".

reduce	A logical value denoting whether to remove age-stages associated solely with 0 transitions. These are only removed in cases where the associated row and column sums in ALL matrices estimated equal 0. Defaults to FALSE.
simple	A logical value indicating whether to produce A, U, and F matrices, or only the latter two. Defaults to FALSE, in which case all three are output.
err_check	A logical value indicating whether to append extra information used in matrix calculation within the output list. 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.
sparse_output	A logical value indicating whether to output matrices in sparse format. Defaults to FALSE, in which case all matrices are output in standard matrix format.

Value

If the user inputs a standard lefkoMod or vrm_input object in argument modelsuite, or individual vital rate models are input separately,then this function will return an object of class lefkoMat. If the user inputs an object of class lefkoModList in argument modelsuite, then the output will be an object of class lefkoMatList, in which each element is an object of class lefkoMat.

A lefkoMat object is a list that holds one full matrix projection model and all of its metadata. The structure has the following elements:

A	A list of full projection matrices in order of sorted patches and occasions. All matrices output in R's matrix class, or in the dgCMatrix class from the Matrix package if sparse.
U	A list of survival transition matrices sorted as in A. All matrices output in R's matrix class, or in the dgCMatrix class from the Matrix package if sparse.
F	A list of fecundity matrices sorted as in A. All matrices output in R's matrix class, or in the dgCMatrix class from the Matrix package if sparse.
hstages	A data frame matrix showing the pairing of ahistorical stages used to create historical stage pairs. Set to NA for age-by-stage MPMs.
agestages	A data frame showing the stage number and stage name corresponding to ahstages, as well as the associated age, of each row in each age-by-stage matrix.
ahstages	A data frame detailing the characteristics of associated ahistorical stages, in the form of a modified stageframe that includes status as an entry stage through reproduction.
labels	A data frame giving the patch and year of each matrix in order. In aflefko2(), only one population may be analyzed at once.
dataqc	A vector showing the numbers of individuals and rows in the vertical dataset used as input.
matrixqc	A short vector describing the number of non-zero elements in U and F matrices, and the number of annual matrices.

modelqc	This is the qc portion of the modelsuite input.
prob_out	An optional element only added if err_check = TRUE. This is a list of vital rate probability matrices, with 7 columns in the order of survival, observation probability, reproduction probability, primary size transition probability, secondary size transition probability, tertiary size transition probability, and probability of juvenile transition to maturity.
allstages	An optional element only added if err_check = TRUE. This is a data frame giving the values used to determine each matrix element capable of being estimated.

Notes

Unlike rlefko2(), rlefko3(), arlefko2(), and rleslie(), this function does not currently distinguish populations. Users wishing to use the same vital rate models across populations should label them as patches (though we do not advise this approach, as populations should typically be treated as statistically independent).

This function will yield incorrect estimates if the models utilized incorporate state in occasion *t*-1. Only use models developed testing for ahistorical effects.

The default behavior of this function is to estimate fecundity with regards to transitions specified via associated fecundity multipliers in the supplement. If this field is left empty, then fecundity will be estimated at full for all transitions leading from reproductive stages to immature and propagule stages.

Stageframes used in this function should include ages for minimum and maximum age for each stage. NAs are treated as 0s in minimum age, and as final_age for maximum age.

Users may at times wish to estimate MPMs using a dataset incorporating multiple patches or subpopulations, but without discriminating between those patches or subpopulations. Should the aim of analysis be a general MPM that does not distinguish these patches or subpopulations, the modelsearch() run should not include patch terms.

Input options including multiple variable names must be entered in the order of variables in occasion *t*+1 and *t*. Rearranging the order will lead to erroneous calculations, and may lead to fatal errors.

Care should be taken to match the random status of year and patch to the states of those variables within the modelsuite. If they do not match, then they will be treated as zeroes in vital rate estimation.

The ipm_method function gives the option of using two different means of estimating the probability of size transition. The midpoint method ("midpoint") refers to the method in which the probability is estimated by first estimating the probability associated with transition from the exact size at the midpoint of the size class using the corresponding probability density function, and then multiplying that value by the bin width of the size class. Doak et al. 2021 (Ecological Monographs) noted that this method can produce biased results, with total size transitions associated with a specific size not totaling to 1.0 and even specific size transition probabilities capable of being estimated at values greater than 1.0. The alternative and default method, "CDF", uses the corresponding cumulative density function to estimate the probability of size transition as the cumulative probability of size transition at the greater limit of the size class minus the cumulative probability of size transition at the lower limit of the size class. The latter method avoids this bias. Note, however, that both methods are exact and unbiased for the Poisson and negative binomial distributions.

Under the Gaussian and gamma size distributions, the number of estimated parameters may differ between the two ipm_method settings. Because the midpoint method has a tendency to incorporate

upward bias in the estimation of size transition probabilities, it is more likely to yield non- zero values when the true probability is extremely close to 0. This will result in the `summary.lefkoMat` function yielding higher numbers of estimated parameters than the `ipm_method = "CDF"` yields in some cases.

Using the `err_check` option will produce a matrix of 7 columns, each characterizing a different vital rate. The product of each row yields an element in the associated `U` matrix. The number and order of elements in each column of this matrix matches the associated matrix in column vector format. Use of this option is generally for the purposes of debugging code.

Individual covariates are treated as categorical only if they are set as random terms. Fixed categorical individual covariates are currently not allowed. However, such terms may be supplied if the `modelsuite` option is set to a `vrm_input` object. In that case, the user should also set the logical random switch for the individual covariate to be used to `TRUE` (e.g., `random.inda = TRUE`).

See Also

`mpm_create()`

`flefko3()`

`flefko2()`

`fleslie()`

`arlefko2()`

`rlefko3()`

`rlefko2()`

`rleslie()`

Examples

```
data(lathyrus)

sizevector <- c(0, 4.6, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 1, 2, 3, 4, 5, 6, 7, 8,
  9)
stagevector <- c("Sd", "Sdl", "Dorm", "Sz1nr", "Sz2nr", "Sz3nr", "Sz4nr",
  "Sz5nr", "Sz6nr", "Sz7nr", "Sz8nr", "Sz9nr", "Sz1r", "Sz2r", "Sz3r",
  "Sz4r", "Sz5r", "Sz6r", "Sz7r", "Sz8r", "Sz9r")
repvector <- c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1)
obsvector <- c(0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
matvector <- c(0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0)
indataset <- c(0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
minima <- c(1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2)
binvec <- c(0, 4.6, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
  0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5)

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

```
lathvertln <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "lnVol88", repstracol = "Intactseed88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframeln,
  stagesize = "sizea", censorcol = "Missing1988", censorkeep = NA,
  NAas0 = TRUE, censor = TRUE)

lathvertln$feca2 <- round(lathvertln$feca2)
lathvertln$feca1 <- round(lathvertln$feca1)
lathvertln$feca3 <- round(lathvertln$feca3)

lathvertln_adults <- subset(lathvertln, stage2index > 2)
surv_model <- glm(alive3 ~ obsage + sizea2 + as.factor(patchid) +
  as.factor(year2), data = lathvertln_adults, family = "binomial")

obs_data <- subset(lathvertln_adults, alive3 == 1)
obs_model <- glm(obsstatus3 ~ obsage + as.factor(patchid) +
  as.factor(year2), data = obs_data, family = "binomial")

size_data <- subset(obs_data, obsstatus3 == 1)
siz_model <- lm(sizea3 ~ sizea2 + repstatus2 + obsage + as.factor(patchid) +
  as.factor(year2), data = size_data)

reps_model <- glm(repstatus3 ~ sizea2 + as.factor(patchid) + as.factor(year2),
  data = size_data, family = "binomial")

fec_data <- subset(lathvertln_adults, repstatus2 == 1)
fec_model <- glm(feca2 ~ sizea2 + obsage + as.factor(patchid) +
  as.factor(year2), data = fec_data, family = "poisson")

lathvertln_juvs <- subset(lathvertln, stage2index < 3)
jsurv_model <- glm(alive3 ~ as.factor(patchid), data = lathvertln_juvs,
  family = "binomial")

jobs_data <- subset(lathvertln_juvs, alive3 == 1)
jobs_model <- glm(obsstatus3 ~ 1, family = "binomial", data = jobs_data)

jsize_data <- subset(jobs_data, obsstatus3 == 1)
jsiz_model <- lm(sizea3 ~ as.factor(year2), data = jsize_data)

jrepst_model <- 0
jmatst_model <- 1

lathsupp2 <- supplemental(stage3 = c("Sd", "Sdl", "mat", "Sd", "Sdl"),
  stage2 = c("Sd", "Sd", "Sdl", "rep", "rep"),
  eststage3 = c(NA, NA, "mat", NA, NA),
  eststage2 = c(NA, NA, "Dorm", NA, NA),
  givenrate = c(0.345, 0.054, NA, NA, NA),
  multiplier = c(NA, NA, 0.8, 0.345, 0.054), type = c(1, 1, 1, 3, 3),
  stageframe = lathframeln, historical = FALSE, agebased = TRUE)
```

```
mod_params <- create_pm(name_terms = TRUE)
mod_params$modelparams[3] <- "patchid"
mod_params$modelparams[5] <- "obsstatus3"
mod_params$modelparams[6] <- "sizea3"
mod_params$modelparams[9] <- "repstatus3"
mod_params$modelparams[11] <- "feca2"
mod_params$modelparams[12] <- "sizea2"
mod_params$modelparams[18] <- "repstatus2"
mod_params$modelparams[22] <- "obsage"

lathmat2age2 <- aflefko2(year = "all", patch = "all", data = lathvertln,
  stageframe = lathframeln, supplement = lathsupp2, final_age = 3,
  surv_model = surv_model, obs_model = obs_model, size_model = siz_model,
  repst_model = reps_model, fec_model = fec_model, jsurv_model = jsurv_model,
  jobs_model = jobs_model, jsize_model = jsiz_model,
  jrepst_model = jrepst_model, jmatst_model = jmatst_model,
  paramnames = mod_params, continue = TRUE, reduce = FALSE)
```

`anthyllis` *Matrix Set of* Anthyllis vulneraria *Populations in Belgium*

Description

A `lefkoMat` object containing projection matrices developed from demographic data gathered on nine *Anthyllis vulneraria* populations from 2003 to 2006 in southwestern Belgium.

Usage

```
data(anthyllis)
```

Format

A `lefkoMat` object holding 27 matrices. The structure of the object is as below:

A The 27 A matrices.

U The 27 survival-transition matrices used to develop the A matrices.

F The 27 fecundity matrices used to develop the A matrices.

hstages Not used, so set to `NA`.

agestages Not used, so set to `NA`.

ahstages The edited stageframe describing the life history of the study organism as interpreted in the original demographic study.

labels The order of the matrices, where each population is treated as a separate patch and each matrix corresponds to a different combination of population and year in time *t*.

matrixqc A vector of integers used in the quality control section of `lefkoMat` summary statements.

dataqc Currently a vector with two `NA` values.

Source

Davison, R. et al. 2010. Demographic effects of extreme weather events on a short-lived calcareous grassland species: stochastic life table response experiments. *Journal of Ecology* 98(2):255-267.

Examples

```
data(anthyllis)

lambda3(anthyllis)
```

---

append_lP *Append Projections To Create New lefkoProj Object*

Description

Function `append_lP()` combines two population projections. It takes two `lefkoProj` objects and appends them into a new `lefkoProj` object.

Usage

```
append_lP(proj1 = NULL, proj2 = NULL)
```

Arguments

`proj1`	A `lefkoProj` object.
`proj2`	A second `lefkoProj` object, based on the same stageframe as `proj1`.

Value

A list of class `lefkoProj`, which always includes the first three elements of the following, and also includes the remaining elements below when a `lefkoMat` object is used as input:

`projection`	A list of lists of matrices showing the total number of individuals per stage per occasion. The first list corresponds to each pop-patch followed by each population (this top-level list is a single element in `f_projection3()`). The inner list corresponds to replicates within each pop-patch or population.
`stage_dist`	A list of lists of the actual stage distribution in each occasion in each replicate in each pop-patch or population.
`rep_value`	A list of lists of the actual reproductive value in each occasion in each replicate in each pop-patch or population.
`pop_size`	A list of matrices showing the total population size in each occasion per replicate (row within data frame) per pop-patch or population (list element). `NA` values will result if projections with different numbers of time steps are appended.
`labels`	A data frame showing the order of populations and patches in item `projection`.
`ahstages`	The original stageframe used in the study.

hstages	A data frame showing the order of historical stage pairs.
agestages	A data frame showing the order of age-stage pairs.
labels	A short data frame indicating the population (always 1), and patch (either the numeric index of the single chosen patch, or 1 in all other cases). Any pop-patches having the same designation across the two input projections will be appended together.
control	A data frame showing the number of replicates and time steps corresponding to each set of projections, where each set corresponds to a pop-patch within the labels object of each input projection.
density	The data frame input under the density option. Only provided if input by the user for at least one of the two projections. Output as a nested list corresponding to each pop-patch - replicate.
density_vr	The data frame input under the density_vr option. Only provided if input by the user for at least one of the two projections. Output as a nested list corresponding to each pop-patch - replicate.

Notes

lefkoProj objects resulting from previous appends can also be appended.

See Also

projection3()

Examples

```
data(cypdata)

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 = 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", "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(0.1, 0.2, 0.2, 0.2, 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_AB <- rlefko2(data = cypraw_v1, stageframe = cypframe_raw,
  year = "all", patch = c("A", "B"), stages = c("stage3", "stage2"),
  size = c("size3added", "size2added"), supplement = cypsupp2r,
  yearcol = "year2",  patchcol = "patchid", indivcol = "individ")

cypmatrix2r_AC <- rlefko2(data = cypraw_v1, stageframe = cypframe_raw,
  year = "all", patch = c("A", "C"), stages = c("stage3", "stage2"),
  size = c("size3added", "size2added"), supplement = cypsupp2r,
  yearcol = "year2",  patchcol = "patchid", indivcol = "individ")

cypproj1 <- projection3(cypmatrix2r_AB, nreps = 5, times = 15,
  stochastic = TRUE)
cypproj2 <- projection3(cypmatrix2r_AC, nreps = 10, times = 20,
  stochastic = TRUE)
cypproj3 <- append_lP(cypproj1, cypproj2)
```

---

arlefko2 *Create Raw Ahistorical Age x Stage Matrix Projection Model*

Description

Function `arlefko2()` returns raw ahistorical age x stage MPMs corresponding to the patches and occasion times given, including the associated component transition and fecundity matrices, data frames detailing the characteristics of ahistorical stages and the exact age-stage combinations corresponding to rows and columns in estimated matrices, and a data frame characterizing the patch and occasion time combinations corresponding to these matrices.

Usage

```
arlefko2(
  data,
  stageframe,
  year = "all",
  pop = NULL,
  patch = NULL,
  censor = FALSE,
  stages = NULL,
```

```
  alive = c("alive3", "alive2"),
  obsst = NULL,
  size = c("sizea3", "sizea2"),
  sizeb = NULL,
  sizec = NULL,
  repst = c("repstatus3", "repstatus2"),
  matst = c("matstatus3", "matstatus2"),
  fec = c("feca3", "feca2"),
  supplement = NULL,
  repmatrix = NULL,
  overwrite = NULL,
  agecol = "obsage",
  yearcol = NULL,
  popcol = NULL,
  patchcol = NULL,
  indivcol = NULL,
  censorcol = NULL,
  censorkeep = 0,
  final_age = NA,
  continue = TRUE,
  prebreeding = TRUE,
  NRasRep = FALSE,
  reduce = FALSE,
  simple = FALSE,
  err_check = FALSE,
  initial_nan = FALSE,
  sparse_output = FALSE
)
```

Arguments

data	A vertical demographic data frame, with variables corresponding to the naming conventions in functions verticalize3() and historicalize3(). Alternatively, a list of bootstrapped data of class hfv_list.
stageframe	A stageframe object that includes information on the size, observation status, propagule status, reproduction status, immaturity status, and maturity status of each ahistorical stage. Should also incorporate bin widths if size is continuous.
year	A variable corresponding to observation occasion, or a set of such values, given in values associated with the year term used in linear model development. Defaults to "all", in which case matrices will be estimated for all occasions.
pop	A variable designating which populations will have matrices estimated. Should be set to specific population names, or to "all" if all populations should have matrices estimated.
patch	A variable designating which patches or subpopulations will have matrices estimated. Should be set to specific patch names, or to "all" if matrices should be estimated for all patches. Defaults to NA, in which case patch designations are ignored.

censor — If TRUE, then data will be removed according to the variable set in censorcol, such that only data with censor values equal to censorkeep will remain. Defaults to FALSE.

stages — An optional vector denoting the names of the variables within the main vertical dataset coding for the stages of each individual in occasions *t*+1 and *t*. The names of stages in these variables should match those used in the stageframe exactly. If left blank, then arlefko2() will attempt to infer stages by matching values of alive, size, repst, and matst to characteristics noted in the associated stageframe.

alive — A vector of names of binomial variables corresponding to status as alive (1) or dead (0) in occasions *t*+1 ans *t*, respectively.

obsst — A vector of names of binomial variables corresponding to observation status in occasions *t*+1, *t*, and *t*-1, respectively. Defaults to NULL, in which case observation status is not used.

size — A vector of names of variables coding the primary size variable in occasions *t*+1 and *t*, respectively. Defaults to c("sizea3", "sizea2").

sizeb — A vector of names of variables coding the secondary size variable in occasions *t*+1 and *t*, respectively. Defaults to NULL, in which case this variable is not used.

sizec — A vector of names of variables coding the tertiary size variable in occasions *t*+1 and *t*, respectively. Defaults to NULL, in which case this variable is not used.

repst — A vector of names of variables coding reproductive status in occasions *t*+1 and *t*, respectively. Defaults to c("repstatus3", "repstatus2"). Must be supplied if stages is not provided.

matst — A vector of names of variables coding maturity status in occasions *t*+1 and *t*, respectively. Defaults to c("matstatus3", "matstatus2"). Must be supplied if stages is not provided.

fec — A vector of names of variables coding fecundity in occasions *t*+1 and *t*, respectively. Defaults to c("feca3", "feca2").

supplement — An optional data frame of class lefkoSD that provides supplemental data that should be incorporated into the MPM. Three kinds of data may be integrated this way: transitions to be estimated via the use of proxy transitions, transition overwrites from the literature or supplemental studies, and transition multipliers for survival and fecundity. This data frame should be produced using the supplemental() function. Can be used in place of or in addition to an overwrite table (see overwrite below) and a reproduction matrix (see repmatrix below).

repmatrix — An optional reproduction matrix. This matrix is composed mostly of 0s, with non-zero entries acting as element identifiers and multipliers for fecundity (with 1 equaling full fecundity). If left blank, and no supplement is provided, then aflefko2() will assume that all stages marked as reproductive produce offspring at 1x that of estimated fecundity, and that offspring production will yield the first stage noted as propagule or immature. To prevent this behavior, input just 0, which will result in fecundity being estimated only for transitions noted in supplement above. Must be the dimensions of an ahistorical stage-based matrix.

overwrite	An optional data frame developed with the overwrite() function describing transitions to be overwritten either with given values or with other estimated transitions. Note that this function supplements overwrite data provided in supplement.
agecol	The variable name or column number coding for age in time *t*.
yearcol	The variable name or column number corresponding to occasion *t* in the dataset.
popcol	The variable name or column number corresponding to the identity of the population.
patchcol	The variable name or column number corresponding to patch in the dataset.
indivcol	The variable name or column number coding individual identity.
censorcol	The variable name or column number denoting the censor status. Only needed if censor = TRUE.
censorkeep	The value of the censor variable denoting data elements to keep. Defaults to 0.
final_age	The final age to model in the matrix, where the first age will be age 0 if post-breeding, and 1 if pre-breeding. Defaults to the maximum age in the dataset.
continue	A logical value designating whether to allow continued survival of individuals past the final age noted in the stageframe, using the demographic characteristics of the final age. Defaults to TRUE.
prebreeding	A logical value indicating whether the life history model is a pre-breeding model. Defaults to TRUE.
NRasRep	If data does not include stage assignments, then this option determines whether non-reproductive and reproductive individuals should be lumped into the same stages. Defaults to FALSE.
reduce	A logical value denoting whether to remove age-stages associated with only zero transitions. These are removed only if the respective row and column sums in ALL matrices estimated equal 0. Defaults to FALSE.
simple	A logical value indicating whether to produce A, U, and F matrices, or only the latter two. Defaults to FALSE, in which case all three are output.
err_check	A logical value indicating whether to append extra information used in matrix calculation within the output list. Defaults to FALSE.
initial_nan	A single logical value indicating whether to initialize matrices was all elements set to NaN. Defaults to FALSE. Cannot be used with sparse matrices.
sparse_output	A logical value indicating whether to output matrices in sparse format. Defaults to FALSE, in which case all matrices are output in standard matrix format.

Value

If the user inputs a standard hfv_data object in argument data, then this function will return an object of class lefkoMat. If the user inputs an object of class hfv_list in argument data, then the output will be an object of class lefkoMatList, in which each element is an object of class lefkoMat.

A lefkoMat object is a list that holds one full matrix projection model and all of its metadata. The structure has the following elements:

A	A list of full projection matrices in order of sorted patches and occasions. All matrices output in R's matrix class, or in the dgCMatrix class from the Matrix package if sparse.
U	A list of survival transition matrices sorted as in A. All matrices output in R's matrix class, or in the dgCMatrix class from the Matrix package if sparse.
F	A list of fecundity matrices sorted as in A. All matrices output in R's matrix class, or in the dgCMatrix class from the Matrix package if sparse.
hstages	A data frame matrix showing the pairing of ahistorical stages used to create historical stage pairs. Set to NA for age-by-stage MPMs.
agestages	A data frame showing the stage number and stage name corresponding to ahstages, as well as the associated age, of each row in each age-by-stage matrix.
ahstages	A data frame detailing the characteristics of associated ahistorical stages, in the form of a modified stageframe that includes status as an entry stage through reproduction.
labels	A data frame giving the patch and year of each matrix in order. In aflefko2(), only one population may be analyzed at once, and so pop = NA
dataqc	A vector showing the numbers of individuals and rows in the vertical dataset used as input.
matrixqc	A short vector describing the number of non-zero elements in U and F matrices, and the number of annual matrices.
modelqc	This is the qc portion of the modelsuite input in function-based MPMs. Empty in this function.

Notes

The default behavior of this function is to estimate fecundity with regards to transitions specified via associated fecundity multipliers in the supplement. If this field is left empty, then fecundity will be estimated at full for all transitions leading from reproductive stages to immature and propagule stages.

Users may at times wish to estimate MPMs using a dataset incorporating multiple patches or subpopulations. Should the aim of analysis be a general MPM that does not distinguish these patches or subpopulations, the patchcol variable should be left to NA, which is the default. Otherwise the variable identifying patch needs to be named.

Input options including multiple variable names must be entered in the order of variables in occasion t+1 and t. Rearranging the order WILL lead to erroneous calculations, and may lead to fatal errors.

Although this function is capable of assigning stages given an input stageframe, it lacks the power of verticalize3() and historicalize3() in this regard. Users are strongly encouraged to use the latter two functions for stage assignment.

See Also

mpm_create()

flefko3()

flefko2()

aflefko2()

fleslie()

rlefko3()

rlefko2()

rleslie()

Examples

```
# Cypripedium example
data(cypdata)

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)
minagevec <- c(1, 1, 2, 3, 4, 5, 5, 5, 5, 5, 5)
maxagevec <- c(rep(NA, 11))

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

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, age_offset = 4)

# Here we use supplemental() to provide overwrite and reproductive info
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, agebased = TRUE)

cyp_mats <- arlefko2(data = cypraw_v1, stageframe = cypframe_raw, year = "all",
  patch = NA, censor = FALSE, stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added"), fec = c("feca3", "feca2"),
```

```
supplement = cypsupp2r, agecol = "obsage", yearcol = "year2",
patchcol = "patchid", indivcol = "individ", prebreeding = TRUE, final_age = NA,
continue = TRUE, reduce = FALSE)
```

---

beverton3 *Two-Parameter Beverton-Holt Function*

Description

Function `beverton3()` creates a vector of values produced by the two- parameter Beverton-Holt function as applied with a user-specified time lag. The two-parameter Beverton-Holt function is given as $\phi_{t+1} = \phi_t \alpha/(1 + \beta n_t)$. Here, if no `separate_N` vector is provided, then $n_t = \phi_t$.

Usage

```
beverton3(
  start_value,
  alpha,
  beta,
  time_steps = 100L,
  time_lag = 1L,
  pre0_subs = FALSE,
  pre0_value = 0,
  substoch = 0L,
  separate_N = NULL
)
```

Arguments

`start_value`	A positive number to start the return vector in time 0.
`alpha`	The alpha parameter in the two-parameter Beverton-Holt function. Must be non-negative.
`beta`	The beta parameter in the two-parameter Beverton-Holt function. Must be non-negative.
`time_steps`	The number of time steps to run the projection. Must be a positive integer.
`time_lag`	A positive integer denoting the number of time steps back for the value of phi in the two-parameter Beverton-Holt function.
`pre0_subs`	A logical value indicating whether to use a number other than that given in `start_value` for values of phi lagged from times prior to time 0.
`pre0_value`	A positive number to use for phi lagged from times prior to time 0. Only used if `pre0_subs = TRUE`.
`substoch`	An integer value indicating the kind of substochasticity to use. Values include: `0`, no substochasticity enforced (the default); `1`, all numbers must be non-negative; and `2`, all numbers should be forced to the interval [0, 1].
`separate_N`	An optional numeric vector with values of N in each time, if phi is to be treated as different from N in the two-parameter model.

Value

A numeric vector of values showing values projected under the two- parameter Beverton-Holt function.

Examples

```
trial_run1 <- beverton3(1, alpha = 0.5, beta = 0.009)
plot(trial_run1)

trial_run2 <- beverton3(1, alpha = 0.5, beta = 0.9)
plot(trial_run2)

trial_run3 <- beverton3(1, alpha = 1, beta = 0.009)
plot(trial_run3)

trial_run4 <- beverton3(1, alpha = 1, beta = 0.9)
plot(trial_run4)

trial_run5 <- beverton3(1, alpha = 5, beta = 0.009)
plot(trial_run5)

trial_run6 <- beverton3(1, alpha = 5, beta = 0.9)
plot(trial_run6)

used_Ns <- c(10, 15, 12, 14, 14, 150, 15, 1, 5, 7, 9, 14, 13, 16, 17, 19,
  25, 26)
trial_run7 <- beverton3(1, alpha = 1, beta = 0.009, separate_N = used_Ns)
plot(trial_run7)
```

---

bootstrap3 *Bootstrap Standardized hfv_data Datasets*

Description

Function `bootstrap3()` takes already standardized `hfvdata` datasets and bootstraps them by individual identity, or by row. All bootstrapping is conducted with replacement.

Usage

```
bootstrap3(
  data,
  by_pop = NULL,
  by_patch = NULL,
  by_indiv = NULL,
  prop_size = NULL,
  max_limit = NULL,
  reps = NULL,
  popcol = NULL,
```

```
  patchcol = NULL,
  indivcol = NULL,
  rename = NULL
)
```

Arguments

data	A data frame of class hfvdata.
by_pop	A logical value indicating whether to sample the data frame by population. If TRUE, then the number of individuals sampled for each population will be set to the respective population's actual number of individuals; otherwise, population identity is ignored. Defaults to TRUE.
by_patch	A logical value indicating whether to sample the data frame by patch. If TRUE, then the number of individuals sampled for each patch will be set to the respective patch's actual number of individuals; otherwise, patch identity is ignored. Defaults to TRUE.
by_indiv	A logical value indicating whether to sample the data frame by individual identity, or by row. If TRUE, then samples by individual identity. Defaults to TRUE.
prop_size	A logical value indicating whether to keep the proportions of individuals (if by_indiv = TRUE) or of rows (if by_indiv = FALSE) in each bootstrapped dataset to the same proportions across populations (if by_pop = TRUE, and patches (if by_patch = TRUE, as in the original dataset. If FALSE, then allows the specific proportions to be set by argument max_limit. Defaults to TRUE.
max_limit	Sets the sample size to pull from the original data frame, if prop_size = FALSE. Defaults to the size of the original dataset if prop_size = TRUE, and to 100 if if prop_size = FALSE. Can also be input as an integer vector giving the number of samples to take by population (if by_pop = TRUE), patch (if by_patch = TRUE), or population-patch (if by_pop = TRUE and by_patch = TRUE).
reps	The number of bootstrap replicates to produce. Defaults to 100.
popcol	A string denoting the variable name coding for population identity in the data frame. Defaults to "popid".
patchcol	A string denoting the variable name coding for patch identity in the data frame. Defaults to "patchid".
indivcol	A string denoting the variable name coding for individual identity in the data frame. Defaults to "individ".
rename	A logical value indicating whether to rename individuals as unique when bootstrapping by individual, even if that individual had been previously chosen. Defaults to FALSE, and can only be set to TRUE if by_indiv = TRUE.

Value

A list of class hfvlist, which is composed of data frames of class hfvdata.

Examples

```
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
matvector <- c(0, 0, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)

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

lathvert <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "Volume88", repstracol = "FCODE88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframe, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, censor = TRUE)

lathboot <- bootstrap3(lathvert, reps = 3)

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "Sd", "Sdl"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "rep", "rep"),
  stage1 = c("Sd", "rep", "Sd", "rep", "all", "all"),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA),
  multiplier = c(NA, NA, NA, NA, 0.345, 0.054),
  type = c(1, 1, 1, 1, 3, 3), type_t12 = c(1, 2, 1, 2, 1, 1),
  stageframe = lathframe, historical = TRUE)

ehrlen3_boot <- rlefko3(data = lathboot, stageframe = lathframe,
  year = c(1989, 1990), stages = c("stage3", "stage2", "stage1"),
  supplement = lathsupp3, yearcol = "year2", indivcol = "individ")
```

---

cond_diff *Extract Conditional Ahistorical Difference Matrices*

Description

Function `cond_diff()` takes a set of historical difference matrices resulting from function `diff_lM()` and decomposes them into ahistorical difference matrices conditional upon stage in time *t*-1.

Usage

```
cond_diff(lDiff, ref = 1L, matchoice = NULL, err_check = NULL)
```

Arguments

lDiff	An object of class lefkoDiff.
ref	Choice of mpm to use as reference. Defaults to 1, which means that the ahstages, hstages, and labels elements for mpm1 will be used for all calculations. Only 1 amd 2 are possible inputs.
matchoice	A character denoting whether to use A, U, or F matrices. Defaults to A matrices.
err_check	A logical value denoting whether to include a data frame of element equivalence from the conditional matrices to the original matrices. Used only for debugging purposes. Defaults to FALSE.

Value

A lefkoCondDiff object, with the following elements:

Mcond	A multi-level list holding the conditional matrices derived from the input lefkoDiff object. The top level of the list corresponds to each historical difference matrix in turn, and the lower level corresponds to each stage in time *t*-1, with individual conditional matrices named for the latter.
hstages	A data frame matrix showing the pairing of ahistorical stages used to create historical stage pairs.
ahstages	A data frame detailing the characteristics of associated ahistorical stages.
labels	A data frame showing the patch and year of each input full A matrix in order.
err_check	An optional data frame showing the order of used element indices to create conditional matrices.

Examples

```
sizevector <- c(0, 0, 0, 0, 0, 0, 1, 3, 6, 11, 19.5)
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, 1.5, 1.5, 3.5, 5)
comments <- c("Dormant seed", "1st yr protocorm", "2nd yr protocorm",
  "3rd yr protocorm", "Seedling", "Dormant adult",
  "Extra small adult (1 shoot)", "Small adult (2-4 shoots)",
  "Medium adult (5-7 shoots)", "Large adult (8-14 shoots)",
  "Extra large adult (>14 shoots)")
cypframe_raw <- sf_create(sizes = sizevector, stagenames = stagevector,
  repstatus = repvector, obsstatus = obsvector, matstatus = matvector,
  propstatus = propvector, immstatus = immvector, indataset = indataset,
  binhalfwidth = binvec, comments = comments)

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)

seeds_per_pod <- 5000

cypsupp2_raw <- supplemental(stage3 = c("SD", "P1", "P2", "P3", "SL", "SL", "D",
    "XSm", "SD", "P1"),
  stage2 = c("SD", "SD", "P1", "P2", "P3", "SL", "SL", "SL", "rep", "rep"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, "D", "XSm", NA, NA),
  eststage2 = c(NA, NA, NA, NA, NA, NA, "XSm", "XSm", NA, NA),
  givenrate = c(0.03, 0.15, 0.1, 0.1, 0.1, 0.05, NA, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, NA, NA, NA, NA, (0.5 * seeds_per_pod),
    (0.5 * seeds_per_pod)),
  type =c(1, 1, 1, 1, 1, 1, 1, 1, 3, 3),
  stageframe = cypframe_raw, historical = FALSE)
cypsupp3_raw <- supplemental(stage3 = c("SD", "SD", "P1", "P1", "P2", "P3", "SL",
    "D", "XSm", "Sm", "D", "XSm", "Sm", "mat", "mat", "mat", "SD", "P1"),
  stage2 = c("SD", "SD", "SD", "SD", "P1", "P2", "P3", "SL", "SL", "SL", "SL",
    "SL", "SL", "D", "XSm", "Sm", "rep", "rep"),
  stage1 = c("SD", "rep", "SD", "rep", "SD", "P1", "P2", "P3", "P3", "P3",
    "SL", "SL", "SL", "SL", "SL", "SL", "mat", "mat"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, NA, "D", "XSm", "Sm", "D", "XSm", "Sm",
    "mat", "mat", "mat", NA, NA),
  eststage2 = c(NA, NA, NA, NA, NA, NA, NA, "XSm", "XSm", "XSm", "XSm", "XSm",
    "XSm", "D", "XSm", "Sm", NA, NA),
  eststage1 = c(NA, NA, NA, NA, NA, NA, NA, "XSm", "XSm", "XSm", "XSm", "XSm",
    "XSm", "XSm", "XSm", "XSm", NA, NA),
  givenrate = c(0.1, 0.1, 0.2, 0.2, 0.2, 0.2, 0.25, NA, NA, NA, NA, NA, NA,
    NA, NA, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
    NA, 0.5, 0.5),
  type = c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3),
  type_t12 = c(1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1),
  stageframe = cypframe_raw, historical = TRUE)

cypmatrix2rp <- rlefko2(data = cypraw_v1, stageframe = cypframe_raw,
  year = "all", patch = "all", stages = c("stage3", "stage2"),
  size = c("size3added", "size2added"), supplement = cypsupp2_raw,
  yearcol = "year2", patchcol = "patchid", indivcol = "individ")

cypmatrix2r <- rlefko2(data = cypraw_v1, stageframe = cypframe_raw,
  year = "all", stages = c("stage3", "stage2"),
  size = c("size3added", "size2added"), supplement = cypsupp2_raw,
  yearcol = "year2", patchcol = "patchid", indivcol = "individ")

cypmatrix3rp <- rlefko3(data = cypraw_v1, stageframe = cypframe_raw,
  year = "all", patch = "all", stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added", "size1added"), supplement = cypsupp3_raw,
  yearcol = "year2", patchcol = "patchid", indivcol = "individ")
```

```
cypmatrix3r <- rlefko3(data = cypraw_v1, stageframe = cypframe_raw,
  year = "all", stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added", "size1added"), supplement = cypsupp3_raw,
  yearcol = "year2", patchcol = "patchid", indivcol = "individ")

cypmatrix2r_3 <- hist_null(cypmatrix2r)
cypmatrix2r_3 <- delete_lM(cypmatrix2r_3, year = 2004)
diff_r <- diff_lM(cypmatrix3r, cypmatrix2r_3)

cypmatrix2rp_3 <- hist_null(cypmatrix2rp)
cypmatrix2rp_3 <- delete_lM(cypmatrix2rp_3, year = 2004)
diff_rp <- diff_lM(cypmatrix3rp, cypmatrix2rp_3)

condr1 <- cond_diff(diff_r, ref = 1)
condr2 <- cond_diff(diff_r, ref = 2)

condrp1 <- cond_diff(diff_rp, matchoice = "U", ref = 1)
condrp2 <- cond_diff(diff_rp, matchoice = "F", ref = 2)
```

`cond_hmpm` *Extract Conditional Ahistorical Matrices from Historical MPM*

Description

Function `cond_hmpm()` takes historical MPMs and decomposes them into ahistorical matrices conditional upon stage in time *t*-1. In effect, the function takes each historical matrix within a lefkoMat object, and forms one ahistorical matrix for each stage in time *t*-1.

Usage

```
cond_hmpm(hmpm, matchoice = NULL, err_check = NULL)
```

Arguments

`hmpm`	A historical matrix projection model of class `lefkoMat`.
`matchoice`	A character denoting whether to use A, U, or F matrices. Defaults to `A` matrices.
`err_check`	A logical value denoting whether to include a data frame of element equivalence from the conditional matrices to the original matrices. Used only for debugging purposes. Defaults to `FALSE`.

Value

A `lefkoCondMat` object, with the following elements:

`Mcond`	A multi-level list holding the conditional A matrices derived from the input `lefkoMat` object. The top level of the list corresponds to each historical matrix in turn, and the lower level corresponds to each stage in time *t*-1, with individual conditional matrices named for the latter.

hstages	A data frame matrix showing the pairing of ahistorical stages used to create historical stage pairs.
ahstages	A data frame detailing the characteristics of associated ahistorical stages.
labels	A data frame showing the patch and year of each input full A matrix in order.
err_check	An optional data frame showing the order of used element indices to create conditional matrices.

Examples

```
data(cypdata)

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 = 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)

cypsupp3r <- supplemental(stage3 = c("SD", "SD", "P1", "P1", "P2", "P3", "SL",
    "D", "XSm", "Sm", "D", "XSm", "Sm", "mat", "mat", "mat", "SD", "P1"),
  stage2 = c("SD", "SD", "SD", "SD", "P1", "P2", "P3", "SL", "SL", "SL", "SL",
    "SL", "SL", "D", "XSm", "Sm", "rep", "rep"),
  stage1 = c("SD", "rep", "SD", "rep", "SD", "P1", "P2", "P3", "P3", "P3",
    "SL", "SL", "SL", "SL", "SL", "SL", "mat", "mat"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, NA, "D", "XSm", "Sm", "D", "XSm", "Sm",
    "mat", "mat", "mat", NA, NA),
  eststage2 = c(NA, NA, NA, NA, NA, NA, NA, "XSm", "XSm", "XSm", "XSm", "XSm",
    "XSm", "D", "XSm", "Sm", NA, NA),
  eststage1 = c(NA, NA, NA, NA, NA, NA, NA, "XSm", "XSm", "XSm", "XSm", "XSm",
    "XSm", "XSm", "XSm", "XSm", NA, NA),
  givenrate = c(0.1, 0.1, 0.2, 0.2, 0.2, 0.2, 0.25, NA, NA, NA, NA, NA, NA,
    NA, NA, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
    NA, 0.5, 0.5),
  type = c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3),
```

```
  type_t12 = c(1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1),
  stageframe = cypframe_raw, historical = TRUE)

cypmatrix3r <- rlefko3(data = cypraw_v1, stageframe = cypframe_raw,
  year = "all", patch = "all", stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added", "size1added"),
  supplement = cypsupp3r, yearcol = "year2", patchcol = "patchid",
  indivcol = "individ")

cypcondmats <- cond_hmpm(cypmatrix3r)
summary(cypcondmats)
```

`create_lM` *Create lefkoMat Object from Given Input Matrices or an MPM Database*

Description

Function `create_lM()` creates lefkoMat objects from supplied matrices and extra information, or from a supplied MPM database such as COMPADRE or COMADRE.

Usage

```
create_lM(
  mats,
  stageframe = NULL,
  hstages = NA,
  agestages = NA,
  historical = FALSE,
  agebystage = FALSE,
  UFdecomp = TRUE,
  entrystage = 1,
  poporder = 1,
  patchorder = 1,
  yearorder = NA,
  matrix_id = NULL,
  add_FC = TRUE,
  sparse_output = FALSE
)
```

Arguments

`mats`	A list of A matrices, or, if importing from a matrix database such as COMPADRE or COMADRE, then the object holding the database.
`stageframe`	A stageframe describing all stages utilized.
`hstages`	A data frame outlining the order of historical stages, if matrices provided in `mats` are historical. Defaults to NA.

agestages	A data frame outlining the order of ahistorical age-stages, if age-by-stage matrices are provided.
historical	A logical value indicating whether input matrices are historical or not. Defaults to FALSE.
agebystage	A logical value indicating whether input matrices are ahistorical age-by-stage matrices. If TRUE, then object agestages is required. Defaults to FALSE.
UFdecomp	A logical value indicating whether U and F matrices should be inferred. Defaults to TRUE.
entrystage	The stage or stages produced by reproductive individuals. Used to determine which transitions are reproductive for U-F decomposition. Defaults to 1, which corresponds to the first stage in the stageframe.
poporder	The order of populations in the list supplied in object mats. Defaults to 1.
patchorder	The order of patches in the list supplied in object mats. Defaults to 1.
yearorder	The order of monitoring occasions in the list supplied in object mats. Defaults to NA, which leads to each matrix within each population-patch combination being a different monitoring occasion.
matrix_id	The values of MatrixID from the used database corresponding to the matrices to import, if importing from a database. Not used if importing a list of matrices.
add_FC	A logical value indicating whether to sum the matF and matC matrices to produce the F matrix. If FALSE, then only uses the matF matrix. Only used if importing from the COMPADRE or COMADRE database. Defaults to TRUE.
sparse_output	A logical value indicating whether to output matrices in sparse format. Defaults to FALSE, in which case all matrices are output in standard matrix format. Does not apply to matrices imported from COMPADRE or COMADRE, which are always in standard format.

Value

A lefkoMat object incorporating the matrices input in object mats as object A, their U and F decompositions in objects U and F (if requested), the provided stageframe as object ahstages, the order of historical stages as object hstages (if historical = TRUE), the order of matrices as object labels, and a short quality control section used by the summary.lefkoMat() function.

Notes for importing lists of matrices

Lists may be composed of a mix of matrices in standard R format (i.e. created via the matrix() function), and in dgCMatrix sparse format (i.e. created via the Matrix::Matrix() function with sparse = TRUE.) All matrices will be forced to one or the other, depending on the value given for the sparse_output argument.

U and F decomposition assumes that elements holding fecundity values are to be interpreted solely as fecundity rates. Users wishing to split these elements between fecundity and survival should do so manually after running this function.

Age-by-stage MPMs require an agestages data frame outlining the order of age-stages. This data frame has 3 variables: stage_id, which is the number of the stage as labelled by the equivalently named variable in the stageframe; stage, which is the official name of the stage as given in the

equivalently named variable in the stageframe; and age, which of course gives the age associated with the stage at that time. The number of rows must be equal to the number of rows and columns of each entered matrix.

Users may edit the dataqc object, setting the first NA to the number of individuals sampled, and the second NA to the number of rows in a vertical version of the demographic dataset. This is not required, however.

Notes for importing from COMPADRE or COMADRE

For this function to operate, users must have either the COMPADRE database or the COMADRE database loaded into the global environment. Note that the sample databases supplied within package Rcompadre will not work with this function.

This function does not and cannot replace the wonderful tools offered to explore the COMPADRE and COMADRE packages. Please see package Rcompadre to use those tools. Note that function import_Com() has no relationship to the Rcompadre development team.

Function import_Com() requires that the dimensions of all matrices imported into a single lefkoMat object be equal.

The reproductive and maturity status of each stage is determined by patterns assessed within the F matrices. Users should check that these values make sense.

Stage names may be edited manually afterward.

Users may edit the dataqc object, setting the first NA to the number of individuals sampled, and the second NA to the number of rows in a vertical version of the demographic dataset. This is not required, however.

See Also

add_lM()

delete_lM()

subset_lM()

Examples

```
# These matrices are of 9 populations of the plant species Anthyllis
# vulneraria, and were originally published in Davison et al. (2010) Journal
# of Ecology 98:255-267 (doi: 10.1111/j.1365-2745.2009.01611.x).

sizevector <- c(1, 1, 2, 3) # These sizes are not from the original paper
stagevector <- c("Sdl", "Veg", "SmFlo", "LFlo")
repvector <- c(0, 0, 1, 1)
obsvector <- c(1, 1, 1, 1)
matvector <- c(0, 1, 1, 1)
immvector <- c(1, 0, 0, 0)
propvector <- c(0, 0, 0, 0)
indataset <- c(1, 1, 1, 1)
binvec <- c(0.5, 0.5, 0.5, 0.5)

anthframe <- sf_create(sizes = sizevector, stagenames = stagevector,
  repstatus = repvector, obsstatus = obsvector, matstatus = matvector,
```

```
  immstatus = immvector, indataset = indataset, binhalfwidth = binvec,
  propstatus = propvector)

# POPN C 2003-2004
XC3 <- matrix(c(0, 0, 1.74, 1.74,
0.208333333, 0, 0, 0.057142857,
0.041666667, 0.076923077, 0, 0,
0.083333333, 0.076923077, 0.066666667, 0.028571429), 4, 4, byrow = TRUE)

# 2004-2005
XC4 <- matrix(c(0, 0, 0.3, 0.6,
0.32183908, 0.142857143, 0, 0,
0.16091954, 0.285714286, 0, 0,
0.252873563, 0.285714286, 0.5, 0.6), 4, 4, byrow = TRUE)

# 2005-2006
XC5 <- matrix(c(0, 0, 0.50625, 0.675,
0, 0, 0, 0.035714286,
0.1, 0.068965517, 0.0625, 0.107142857,
0.3, 0.137931034, 0, 0.071428571), 4, 4, byrow = TRUE)

# POPN E 2003-2004
XE3 <- matrix(c(0, 0, 2.44, 6.569230769,
0.196428571, 0, 0, 0,
0.125, 0.5, 0, 0,
0.160714286, 0.5, 0.133333333, 0.076923077), 4, 4, byrow = TRUE)

XE4 <- matrix(c(0, 0, 0.45, 0.646153846,
0.06557377, 0.090909091, 0.125, 0,
0.032786885, 0, 0.125, 0.076923077,
0.049180328, 0, 0.125, 0.230769231), 4, 4, byrow = TRUE)

XE5 <- matrix(c(0, 0, 2.85, 3.99,
0.083333333, 0, 0, 0,
0, 0, 0, 0,
0.416666667, 0.1, 0, 0.1), 4, 4, byrow = TRUE)

mats_list <- list(XC3, XC4, XC5, XE3, XE4, XE5)
yr_ord <- c(1, 2, 3, 1, 2, 3)
pch_ord <- c(1, 1, 1, 2, 2, 2)

anth_lefkoMat <- create_lM(mats_list, anthframe, hstages = NA,
  historical = FALSE, poporder = 1, patchorder = pch_ord, yearorder = yr_ord)

# A theoretical example showcasing historical matrices
sizevector <- c(1, 2, 3) # These sizes are not from the original paper
stagevector <- c("Sdl", "Veg", "Flo")
repvector <- c(0, 0, 1)
obsvector <- c(1, 1, 1)
matvector <- c(0, 1, 1)
immvector <- c(1, 0, 0)
propvector <- c(1, 0, 0)
indataset <- c(1, 1, 1)
```

```
binvec <- c(0.5, 0.5, 0.5)

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

A1 <- matrix(c(0.10, 0, 0, 0.12, 0, 0, 0.15, 0, 0,
  0.15, 0, 0, 0.17, 0, 0, 0.20, 0, 0,
  0.20, 0, 0, 0.22, 0, 0, 0.25, 0, 0,
  0, 0.20, 0, 0, 0.22, 0, 0, 0.25, 0,
  0, 0.25, 0, 0, 0.27, 0, 0, 0.30, 0,
  0, 0.30, 0, 0, 0.32, 0, 0, 0.35, 0,
  0, 0, 2.00, 0, 0, 3.00, 0, 0, 4.00,
  0, 0, 0.35, 0, 0, 0.37, 0, 0, 0.40,
  0, 0, 0.40, 0, 0, 0.42, 0, 0, 0.45), 9, 9, byrow = TRUE)

A2 <- matrix(c(0.10, 0, 0, 0.12, 0, 0, 0.15, 0, 0,
  0.15, 0, 0, 0.17, 0, 0, 0.20, 0, 0,
  0.20, 0, 0, 0.22, 0, 0, 0.25, 0, 0,
  0, 0.20, 0, 0, 0.22, 0, 0, 0.25, 0,
  0, 0.25, 0, 0, 0.27, 0, 0, 0.30, 0,
  0, 0.30, 0, 0, 0.32, 0, 0, 0.35, 0,
  0, 0, 5.00, 0, 0, 6.00, 0, 0, 7.00,
  0, 0, 0.35, 0, 0, 0.37, 0, 0, 0.40,
  0, 0, 0.40, 0, 0, 0.42, 0, 0, 0.45), 9, 9, byrow = TRUE)

A3 <- matrix(c(0.10, 0, 0, 0.12, 0, 0, 0.15, 0, 0,
  0.15, 0, 0, 0.17, 0, 0, 0.20, 0, 0,
  0.20, 0, 0, 0.22, 0, 0, 0.25, 0, 0,
  0, 0.20, 0, 0, 0.22, 0, 0, 0.25, 0,
  0, 0.25, 0, 0, 0.27, 0, 0, 0.30, 0,
  0, 0.30, 0, 0, 0.32, 0, 0, 0.35, 0,
  0, 0, 8.00, 0, 0, 9.00, 0, 0, 10.00,
  0, 0, 0.35, 0, 0, 0.37, 0, 0, 0.40,
  0, 0, 0.40, 0, 0, 0.42, 0, 0, 0.45), 9, 9, byrow = TRUE)

B1 <- matrix(c(0.10, 0, 0, 0.12, 0, 0, 0.15, 0, 0,
  0.15, 0, 0, 0.17, 0, 0, 0.20, 0, 0,
  0.20, 0, 0, 0.22, 0, 0, 0.25, 0, 0,
  0, 0.20, 0, 0, 0.22, 0, 0, 0.25, 0,
  0, 0.25, 0, 0, 0.27, 0, 0, 0.30, 0,
  0, 0.30, 0, 0, 0.32, 0, 0, 0.35, 0,
  0, 0, 11.00, 0, 0, 12.00, 0, 0, 13.00,
  0, 0, 0.35, 0, 0, 0.37, 0, 0, 0.40,
  0, 0, 0.40, 0, 0, 0.42, 0, 0, 0.45), 9, 9, byrow = TRUE)

B2 <- matrix(c(0.10, 0, 0, 0.12, 0, 0, 0.15, 0, 0,
  0.15, 0, 0, 0.17, 0, 0, 0.20, 0, 0,
  0.20, 0, 0, 0.22, 0, 0, 0.25, 0, 0,
  0, 0.20, 0, 0, 0.22, 0, 0, 0.25, 0,
  0, 0.25, 0, 0, 0.27, 0, 0, 0.30, 0,
  0, 0.30, 0, 0, 0.32, 0, 0, 0.35, 0,
```

```
  0, 0, 14.00, 0, 0, 15.00, 0, 0, 16.00,
  0, 0, 0.35, 0, 0, 0.37, 0, 0, 0.40,
  0, 0, 0.40, 0, 0, 0.42, 0, 0, 0.45), 9, 9, byrow = TRUE)

B3 <- matrix(c(0.10, 0, 0, 0.12, 0, 0, 0.15, 0, 0,
  0.15, 0, 0, 0.17, 0, 0, 0.20, 0, 0,
  0.20, 0, 0, 0.22, 0, 0, 0.25, 0, 0,
  0, 0.20, 0, 0, 0.22, 0, 0, 0.25, 0,
  0, 0.25, 0, 0, 0.27, 0, 0, 0.30, 0,
  0, 0.30, 0, 0, 0.32, 0, 0, 0.35, 0,
  0, 0, 17.00, 0, 0, 18.00, 0, 0, 19.00,
  0, 0, 0.35, 0, 0, 0.37, 0, 0, 0.40,
  0, 0, 0.40, 0, 0, 0.42, 0, 0, 0.45), 9, 9, byrow = TRUE)

histmats <- list(A1, A2, A3, B1, B2, B3)
stageframe <- exframe
pch_ord <- c("A", "A", "A", "B", "B", "B")
yr_ord <- c(1, 2, 3, 1, 2, 3)

hist_trial <- create_lM(histmats, exframe, historical = TRUE,
  UFdecomp = TRUE, entrystage = 1, patchorder = pch_ord, yearorder = yr_ord)
```

`create_pm` *Creates a Skeleton Paramnames Object for Use in Function-based Modeling*

Description

Creates a simple skeleton `paramnames` object that can be entered as input in functions `flefko2()`, `flefko3()`, and `aflefko2()`.

Usage

```
create_pm(name_terms = FALSE)
```

Arguments

`name_terms`	A logical value indicating whether to start each variable name as none if `FALSE`, or as the default `modelparams` name if `TRUE`. Defaults to `FALSE`.

Value

A three column data frame, of which the first describes the parameters in reasonably plain English, the second gives the name of the parameter within the MPM generating functions, and the third is to be edited with the names of the variables as they appear in the models.

Notes

The third column in the resulting object should be edited with the names only of those variables actually used in vital rate modeling. This `paramnames` object should apply to all models used in a single MPM building exercise. So, for example, if the models used include random terms, then they should all have the same random terms. Fixed terms can vary, however.

Examples

```
our_pm <- create_pm()
our_pm
```

`cycle_check` *Check Continuity of Life Cycle through Matrices in lefkoMat Objects*

Description

Function `cycle_check()` tests whether stages, stage-pairs, or age-stages connect in matrices within `lefkoMat` objects.

Usage

```
cycle_check(mpm, quiet = NULL)
```

Arguments

`mpm`	An object of class lefkoMat, a matrix, or a list of matrices.
`quiet`	A logical variable indicating whether to suppress diagnostic messages. Defaults to `FALSE`.

Value

Returns a list with two elements, both of which are also lists. The first list, `no_in`, contains as many elements as matrices, with each element containing an integer vector showing the identification numbers of stages, stage-pairs, or age-stages, in each matrix that do not show any transitions leading to them. The second list, `no_out`, is structured similarly to the first, but shows stages, stage-pairs, or age-stages from which there are no transitions leading out.

Notes

This function tests whether stages, stage-pairs, and age-stages are connected to others in matrices used for projection. Whether stages, stage-pairs, or age-stages are shown depends on whether the MPM is ahistorical / age-based, historical stage-based, or age-by-stage, respectively. Checks are performed by testing whether each column in a matrix includes non-zero transitions to other columns, and by testing whether any columns have no transitions to them from other columns. If any such columns are found, then function `cycle_check` will export an integer vector giving the column numbers with problems. These column numbers may then be checked against the `stage_id`

column of the associated stageframe in the case of a ahistorical or age-based MPM, against the row number of the associated hstages data frame in the case of a historical MPM, or against the row number of the associated agestages data frame in the case of an age-by-stage MPM.

Examples

```
data(cypdata)

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 = 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", "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", patch = "all", stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added"), supplement = cypsupp2r,
  yearcol = "year2", patchcol = "patchid", indivcol = "individ")

cycle_check(cypmatrix2r)
```

cypdata *Demographic Dataset of* Cypripedium candidum *Population, in Horizontal Format*

Description

A dataset containing the states and fates of *Cypripedium candidum* (white lady's slipper orchids), family Orchidaceae, from a population in Illinois, USA, resulting from monitoring that occurred annually between 2004 and 2009.

Usage

```
data(cypdata)
```

Format

A data frame with 77 individuals and 29 variables. Each row corresponds to an unique individual, and each variable from `size.04` on refers to the state of the individual in a particular year.

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

patch A variable refering to patch within the population.

X An X coordinate for the plant within the population.

Y A Y coordinate for the plant within the population.

censor A variable coding for whether the data point is valid. An entry of 1 means that it is so.

Inf2.04 Number of double inflorescences in 2004.

Inf.04 Number of inflorescences in 2004.

Veg.04 Number of stems without inflorescences in 2004.

Pod.04 Number of fruits in 2004.

Inf2.05 Number of double inflorescences in 2005.

Inf.05 Number of inflorescences in 2005.

Veg.05 Number of stems without inflorescences in 2005.

Pod.05 Number of fruits in 2005.

Inf2.06 Number of double inflorescences in 2006.

Inf.06 Number of inflorescences in 2006.

Veg.06 Number of stems without inflorescences in 2006.

Pod.06 Number of fruits in 2006.

Inf2.07 Number of double inflorescences in 2007.

Inf.07 Number of inflorescences in 2007.

Veg.07 Number of stems without inflorescences in 2007.

Pod.07 Number of fruits in 2007.

Inf2.08 Number of double inflorescences in 2008.

Inf.08 Number of inflorescences in 2008.

Veg.08 Number of stems without inflorescences in 2008.

Pod.08 Number of fruits in 2008.

Inf2.09 Number of double inflorescences in 2009.

Inf.09 Number of inflorescences in 2009.

Veg.09 Number of stems without inflorescences in 2009.

Pod.09 Number of fruits in 2009.

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

```
data(cypdata)

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 = 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", "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", patch = "all", stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added"), supplement = cypsupp2r,
  yearcol = "year2", patchcol = "patchid", indivcol = "individ")

lambda3(cypmatrix2r)
```

`cypvert` *Demographic Dataset of* Cypripedium candidum *Population, in Vertical Format*

Description

A dataset containing the states and fates of *Cypripedium candidum* (white lady's slipper orchids), family Orchidaceae, from a population in Illinois, USA, resulting from monitoring that occurred annually between 2004 and 2009. Same dataset as `cypdata`, but arranged in an ahistorical vertical format.

Usage

```
data(cypvert)
```

Format

A data frame with 77 individuals, 322 rows, and 14 variables. Each row corresponds to a specific two-year transition for a specific individual. Variable codes are similar to those for `cypdata`, but use `.2` to identify occasion *t* and `.3` to identify occasion *t*+1.

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

patch A variable refering to patch within the population.

X An X coordinate for the plant within the population.

Y A Y coordinate for the plant within the population.

censor A variable coding for whether the data point is valid. An entry of 1 means that it is so.

year2 Year in occasion *t*.

Inf2.2 Number of double inflorescences in occasion *t*.

Inf.2 Number of inflorescences in occasion *t*.

Veg.2 Number of stems without inflorescences in occasion *t*.

Pod.2 Number of fruits in occasion *t*.

Inf2.3 Number of double inflorescences in occasion *t*+1.

Inf.3 Number of inflorescences in occasion *t*+1.

Veg.3 Number of stems without inflorescences in occasion *t*+1.

Pod.3 Number of fruits in occasion *t*+1.

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

```
data(cypvert)

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_v2 <- historicalize3(data = cypvert, patchidcol = "patch",
  individcol = "plantid", year2col = "year2", sizea2col = "Inf2.2",
  sizea3col = "Inf2.3", sizeb2col = "Inf.2", sizeb3col = "Inf.3",
  sizec2col = "Veg.2", sizec3col = "Veg.3", repstra2col = "Inf2.2",
  repstra3col = "Inf2.3", repstrb2col = "Inf.2", repstrb3col = "Inf.3",
  feca2col = "Pod.2", feca3col = "Pod.3", repstrrel = 2,
  stageassign = cypframe_raw, stagesize = "sizeadded", censorcol = "censor",
  censor = FALSE, NAas0 = TRUE, NRasRep = TRUE, reduce = 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_v2, stageframe = cypframe_raw,
  year = "all", patch = "all", stages = c("stage3", "stage2"),
  size = c("size3added", "size2added"), supplement = cypsupp2r,
  yearcol = "year2", patchcol = "patchid", indivcol = "individ")

lambda3(cypmatrix2r)
```

delete_lM *Delete Matrices from lefkoMat or lefkoMatList Object*

Description

Function delete_lM() deletes matrices from lefkoMat and lefkoMatList objects.

Usage

```
delete_lM(lM, mat_num = NA, pop = NA, patch = NA, year = NA)
```

Arguments

lM	The lefkoMat or lefkoMatList object to delete matrices from.
mat_num	Either a single integer corresponding to the matrix to remove within the labels element of lM, or a vector of such integers.
pop	The population designation for matrices to remove. Only used if mat_num is not given.
patch	The patch designation for matrices to remove. Only used if mat_num is not given.
year	The time *t* designation for matrices to remove. Only used if mat_num is not given.

Value

A lefkoMat or lefkoMatList object in which the matrices specified in lM have been removed. Note that, if applying to a lefkoMatList object, then matrices will be deleted from ALL composite lefkoMat objects.

Notes

If mat_num is not provided, then at least one of pop, patch, or year must be provided. If at least two of pop, patch, and year are provided, then function detele_lM() will identify matrices to remove as the intersection of provided inputs.

See Also

create_lM()

add_lM()

subset_lM()

Examples

```
# These matrices are of 9 populations of the plant species Anthyllis
# vulneraria, and were originally published in Davison et al. (2010) Journal
# of Ecology 98:255-267 (doi: 10.1111/j.1365-2745.2009.01611.x).

sizevector <- c(1, 1, 2, 3) # These sizes are not from the original paper
stagevector <- c("Sdl", "Veg", "SmFlo", "LFlo")
repvector <- c(0, 0, 1, 1)
obsvector <- c(1, 1, 1, 1)
matvector <- c(0, 1, 1, 1)
immvector <- c(1, 0, 0, 0)
propvector <- c(0, 0, 0, 0)
indataset <- c(1, 1, 1, 1)
binvec <- c(0.5, 0.5, 0.5, 0.5)

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

# POPN C 2003-2004
XC3 <- matrix(c(0, 0, 1.74, 1.74,
0.208333333, 0, 0, 0.057142857,
0.041666667, 0.076923077, 0, 0,
0.083333333, 0.076923077, 0.066666667, 0.028571429), 4, 4, byrow = TRUE)

# 2004-2005
XC4 <- matrix(c(0, 0, 0.3, 0.6,
0.32183908, 0.142857143, 0, 0,
0.16091954, 0.285714286, 0, 0,
0.252873563, 0.285714286, 0.5, 0.6), 4, 4, byrow = TRUE)

# 2005-2006
XC5 <- matrix(c(0, 0, 0.50625, 0.675,
0, 0, 0, 0.035714286,
0.1, 0.068965517, 0.0625, 0.107142857,
0.3, 0.137931034, 0, 0.071428571), 4, 4, byrow = TRUE)

# POPN E 2003-2004
XE3 <- matrix(c(0, 0, 2.44, 6.569230769,
0.196428571, 0, 0, 0,
0.125, 0.5, 0, 0,
0.160714286, 0.5, 0.133333333, 0.076923077), 4, 4, byrow = TRUE)

XE4 <- matrix(c(0, 0, 0.45, 0.646153846,
0.06557377, 0.090909091, 0.125, 0,
0.032786885, 0, 0.125, 0.076923077,
0.049180328, 0, 0.125, 0.230769231), 4, 4, byrow = TRUE)

XE5 <- matrix(c(0, 0, 2.85, 3.99,
0.083333333, 0, 0, 0,
0, 0, 0, 0,
```

```
0.416666667, 0.1, 0, 0.1), 4, 4, byrow = TRUE)

mats_list <- list(XC3, XC4, XC5, XE3, XE4, XE5)
yr_ord <- c(1, 2, 3, 1, 2, 3)
pch_ord <- c(1, 1, 1, 2, 2, 2)

anth_lefkoMat <- create_lM(mats_list, anthframe, hstages = NA,
  historical = FALSE, poporder = 1, patchorder = pch_ord, yearorder = yr_ord)

smaller_anth_lM <- delete_lM(anth_lefkoMat, patch = 2)
```

`density_input` *Set Density Dependence Relationships in Matrix Elements*

Description

Function `density_input()` provides all necessary data to incorporate density dependence into a `lefkoMat` object, a list of matrices, or a single matrix. Five forms of density dependence are allowed, including the Ricker function, the Beverton-Holt function, the Usher function, the logistic function, and the additive limit function. In each case, density must have an effect with a delay of at least one time-step (see Notes). The resulting data frame provides a guide for other `lefko3` functions to modify matrix elements by density.

Usage

```
density_input(
  mpm,
  stage3 = NULL,
  stage2 = NULL,
  stage1 = NULL,
  age2 = NULL,
  style = NULL,
  time_delay = NULL,
  alpha = NULL,
  beta = NULL,
  gamma = NULL,
  type = NULL,
  type_t12 = NULL
)
```

Arguments

`mpm`	The `lefkoMat` object that will be subject to density dependent projection.
`stage3`	A vector showing the name or number of the stage in occasion *t*+1 in the transitions to be affected by density. Abbreviations for groups of stages are also usable (see Notes).

`stage2` A vector showing the name or number of the stage in occasion *t* in the transition to be affected by density. Abbreviations for groups of stages are also usable (see Notes).

`stage1` A vector showing the name or number of the stage in occasion *t*-1 in the transition to be affected by density. Only needed if a historical MPM is used. Abbreviations for groups of stages are also usable (see Notes).

`age2` A vector showing the age of the stage in occasion *t* in the transition to be affected by density. Only needed if an age-by-stage MPM is used.

`style` A vector coding for the style of density dependence on each transition subject to density dependence. Options include `1`, `ricker`, `ric`, or `r` for the Ricker function; `2`, `beverton`, `bev`, and `b` for the Beverton-Holt function; `3`, `usher`, `ush`, and `u` for the Usher function; `4`, `logistic`, `log`, and `l` for the logistic function; and `5`, `additive`, `add`, and `a` for the additive limit function. If only a single code is provided, then all noted transitions are assumed to be subject to this style of density dependence. Defaults to `ricker`.

`time_delay` An integer vector indicating the number of occasions back on which density dependence operates. Defaults to `1`, and may not equal any integer less than `1`. If a single number is input, then all noted transitions are assumed to be subject to this time delay. Does not apply to the additive limit function, which uses only the current population.

`alpha` A vector indicating the numeric values to use as the alpha term in the two parameter Ricker, Beverton-Holt, or Usher function, or the value of the carrying capacity *K* to use in the logistic or additive limit functions (see `Notes` section for more on this term). If a single number is provided, then all noted transitions are assumed to be subject to this value of alpha. Defaults to `1`.

`beta` A vector indicating the numeric values to use as the beta term in the two parameter Ricker, Beverton-Holt, or Usher function, or the multiplier on the previous population size in the additive limit function. Also used to indicate whether to use *K* as a hard limit in the logistic equation (see section `Notes` below). If a single number is provided, then all noted transitions are assumed to be subject to this value of `beta`. Defaults to `1`.

`gamma` A vector indicating the numeric values to use as the gamma term in any function using a third term. Currently, this is only used in the additive limit function, and denotes the minimum number of individuals allowed in a particular stage.

`type` A 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; or `2`, `F`, or `f` for the replacement of a fecundity transition. If empty or not provided, then defaults to `1` for survival transition.

`type_t12` An optional 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.

Value

A data frame of class `lefkoDens`. This object can be used as input in function `projection3()`.

Variables in this object include the following:

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, if applicable.
age2	Age at occasion *t* in the transition to be replaced, if applicable.
style	Style of density dependence, coded as 1, 2, 3, 4, or 5 for the Ricker, Beverton-Holt, Usher, logistic, or additive limit function, respectively.
time_delay	The time delay on density dependence, in time steps.
alpha	The value of alpha in the Ricker, Beverton-Holt, or Usher function, or the value of carrying capacity, *K*, in the logistic or additive limit functions.
beta	The value of beta in the Ricker, Beverton-Holt, or Usher function, or the value of the multiplier in the additive limit function.
gamma	The value of gamma, if such a value exists, as in the additive limit function.
type	Designates whether the transition from occasion *t* to occasion *t*+1 is a survival transition probability (1), or a fecundity rate (2).
type_t12	Designates whether the transition from occasion *t*-1 to occasion *t* is a survival transition probability (1), a fecundity rate (2).

Notes

This function provides inputs when density dependence is operationalized directly on matrix elements. It can be used in both `projection3()` and `f_projection3()`. Users wishing to modify vital rate functions by density dependence functions for use in function-based projections with function `f_projection3()` should use function `density_vr()` to provide the correct inputs.

The parameters `alpha` and `beta` are applied according to the two-parameter Ricker function, the two-parameter Beverton-Holt function, the two-parameter Usher function, the one-parameter logistic function, or the additive limit function. Although the default is that a 1 time step delay is assumed, greater time delays can be set through the `time_delay` option.

The gamma term is currently only used for the additive limit function, and designates a minimum number of individuals in a particular stage, if other than 0. If used, then the limit will be applied to the stage given in `stage3`.

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.

When using the logistic function, it is possible that the time delay used in density dependent simulations will cause matrix elements to become negative. To prevent this behavior, set the associated `beta` term to `1.0`. Doing so will set `K` as the hard limit in the logistic equation, essentially setting a minimum limit at `0` for all matrix elements modified.

See Also

start_input()

projection3()

Examples

```
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
matvector <- c(0, 0, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)

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

lathvert <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "Volume88", repstracol = "FCODE88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframe, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, censor = TRUE)

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "Sd", "Sdl", "mat"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "rep", "rep", "Sdl"),
  stage1 = c("Sd", "rep", "Sd", "rep", "npr", "npr", "Sd"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, "mat"),
  eststage2 = c(NA, NA, NA, NA, NA, NA, "Sdl"),
  eststage1 = c(NA, NA, NA, NA, NA, NA, "NotAlive"),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, 0.345, 0.054, NA),
  type = c(1, 1, 1, 1, 3, 3, 1), type_t12 = c(1, 2, 1, 2, 1, 1, 1),
  stageframe = lathframe, historical = TRUE)

ehrlen3 <- rlefko3(data = lathvert, stageframe = lathframe, year = "all",
  stages = c("stage3", "stage2", "stage1"), supplement = lathsupp3,
  yearcol = "year2", indivcol = "individ")

ehrlen3mean <- lmean(ehrlen3)

e3d <- density_input(ehrlen3mean, stage3 = c("Sd", "Sdl"),
  stage2 = c("rep", "rep"), stage1 = c("all", "all"), style = 1,
  time_delay = 1, alpha = 1, beta = 0, type = c(2, 2), type_t12 = c(1, 1))

lathproj <- projection3(ehrlen3, nreps = 5, stochastic = TRUE, substoch = 2,
```

```
  density = e3d)
```

---

density_vr *Set Density Dependence Relationships in Vital Rates*

Description

Function density_vr() provides all necessary data to incorporate density dependence into the vital rate functions used to create matrices in function-based projections using function f_projection3(). Four forms of density dependence are allowed, including the Ricker function, the Beverton-Holt function, the Usher function, and the logistic function. In each case, density must have an effect with at least a one time-step delay (see Notes).

Usage

```
density_vr(
 density_yn = c(FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE,
    FALSE, FALSE, FALSE, FALSE),
  style = c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
  time_delay = c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1),
  alpha = c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
  beta = c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
)
```

Arguments

density_yn	A 14 element logical vector denoting whether each vital rate is subject to density dependence. The order of vital rates is: survival probability, observation probability, primary size transition, secondary size transition, tertiary size transition, reproductive status probability, fecundity rate, juvenile survival probability, juvenile observation probability, juvenile primary size transition, juvenile secondary size transition, juvenile tertiary size transition, juvenile reproductive status probability, and juvenile maturity status probability. Defaults to a vector of 14 FALSE values.
style	A 14 element vector coding for the style of density dependence on each vital rate. Options include 0: no density dependence, 1, ricker, ric, or r for the Ricker function; 2, beverton, bev, and b for the Beverton-Holt function; 3, usher, ush, and u for the Usher function; and 4, logistic, log, and l for the logistic function. Defaults to 14 values of 0.
time_delay	A 14 element vector indicating the number of occasions back on which density dependence operates. Defaults to 14 values of 1, and may not include any number less than 1.
alpha	A 14 element vector indicating the numeric values to use as the alpha term in the two parameter Ricker, Beverton-Holt, or Usher function, or the value of the carrying capacity K to use in the logistic equation (see Notes for more on this term). Defaults to 14 values of 0.

`beta` A 14 element vector indicating the numeric values to use as the beta term in the two parameter Ricker, Beverton-Holt, or Usher function. Used to indicate whether to use *K* as a hard limit in the logistic equation (see `Notes` below). Defaults to 14 values of `0`.

Value

A data frame of class `lefkoDensVR` with 14 rows, one for each vital rate in the order of: survival probability, observation probability, primary size transition, secondary size transition, tertiary size transition, reproductive status probability, fecundity rate, juvenile survival probability, juvenile observation probability, juvenile primary size transition, juvenile secondary size transition, juvenile tertiary size transition, juvenile reproductive status probability, and juvenile maturity status probability. This object can be used as input in function `f_projection3()`.

Variables in this object include the following:

`vital_rate` The vital rate to be modified.

`density_yn` Logical value indicating whether vital rate will be subject to density dependence.

`style` Style of density dependence, coded as `1`, `2`, `3`, `4`, or `0` for the Ricker, Beverton-Holt, Usher, or logistic function, or no density dependence, respectively.

`time_delay` The time delay on density dependence, in time steps.

`alpha` The value of alpha in the Ricker, Beverton-Holt, or Usher function, or the value of carrying capacity, *K*, in the logistic function.

`beta` The value of beta in the Ricker, Beverton-Holt, or Usher function.

Notes

This function provides inputs when density dependence is operationalized directly on vital rates. It can be used only in function `f_projection3()`. Users wishing to modify matrix elements directly by density dependence functions for use in function-based or raw projections with functions `projection3()` and `f_projection3()` should use function `density_input()` to provide the correct inputs.

The parameters `alpha` and `beta` are applied according to the two-parameter Ricker function, the two-parameter Beverton-Holt function, the two-parameter Usher function, or the one-parameter logistic function. Although the default is that a 1 time step delay is assumed, greater time delays can be set through the `time_delay` option.

When using the logistic function, it is possible that the time delay used in density dependent simulations will cause matrix elements to become negative. To prevent this behavior, set the associated `beta` term to `1.0`. Doing so will set `K` as the hard limit in the logistic equation, essentially setting a minimum limit at `0` for all matrix elements modified.

See Also

`density_input()`

`f_projection3()`

Examples

```
data(lathyrus)

sizevector <- c(0, 4.6, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 1, 2, 3, 4, 5, 6, 7, 8,
  9)
stagevector <- c("Sd", "Sdl", "Dorm", "Sz1nr", "Sz2nr", "Sz3nr", "Sz4nr",
  "Sz5nr", "Sz6nr", "Sz7nr", "Sz8nr", "Sz9nr", "Sz1r", "Sz2r", "Sz3r",
  "Sz4r", "Sz5r", "Sz6r", "Sz7r", "Sz8r", "Sz9r")
repvector <- c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1)
obsvector <- c(0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
matvector <- c(0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0)
indataset <- c(0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 4.6, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
  0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5)

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

lathvertln <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "lnVol88", repstracol = "Intactseed88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframeln, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, NAas0 = TRUE, censor = TRUE)

lathvertln$feca2 <- round(lathvertln$feca2)
lathvertln$feca1 <- round(lathvertln$feca1)
lathvertln$feca3 <- round(lathvertln$feca3)

lathvertln_adults <- subset(lathvertln, stage2index > 2)
surv_model <- glm(alive3 ~ sizea2 + sizea1 + as.factor(patchid) +
  as.factor(year2), data = lathvertln_adults, family = "binomial")

obs_data <- subset(lathvertln_adults, alive3 == 1)
obs_model <- glm(obsstatus3 ~ as.factor(patchid), data = obs_data,
  family = "binomial")

size_data <- subset(obs_data, obsstatus3 == 1)
siz_model <- lm(sizea3 ~ sizea2 + sizea1 + repstatus1 + as.factor(patchid) +
  as.factor(year2), data = size_data)

reps_model <- glm(repstatus3 ~ sizea2 + sizea1 + as.factor(patchid) +
  as.factor(year2), data = size_data, family = "binomial")

fec_data <- subset(lathvertln_adults, repstatus2 == 1)
fec_model <- glm(feca2 ~ sizea2 + sizea1 + repstatus1 + as.factor(patchid),
  data = fec_data, family = "poisson")
```

```
lathvertln_juvs <- subset(lathvertln, stage2index < 3)
jsurv_model <- glm(alive3 ~ as.factor(patchid), data = lathvertln_juvs,
  family = "binomial")

jobs_data <- subset(lathvertln_juvs, alive3 == 1)
jobs_model <- glm(obsstatus3 ~ 1, family = "binomial", data = jobs_data)

jsize_data <- subset(jobs_data, obsstatus3 == 1)
jsiz_model <- lm(sizea3 ~ as.factor(year2), data = jsize_data)

jrepst_model <- 0
jmatst_model <- 1

mod_params <- create_pm(name_terms = TRUE)
mod_params$modelparams[3] <- "patchid"
mod_params$modelparams[4] <- "alive3"
mod_params$modelparams[5] <- "obsstatus3"
mod_params$modelparams[6] <- "sizea3"
mod_params$modelparams[9] <- "repstatus3"
mod_params$modelparams[11] <- "feca2"
mod_params$modelparams[12] <- "sizea2"
mod_params$modelparams[13] <- "sizea1"
mod_params$modelparams[18] <- "repstatus2"
mod_params$modelparams[19] <- "repstatus1"

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "mat", "Sd", "Sdl"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "Sdl", "rep", "rep"),
  stage1 = c("Sd", "rep", "Sd", "rep", "Sd", "mat", "mat"),
  eststage3 = c(NA, NA, NA, NA, "mat", NA, NA),
  eststage2 = c(NA, NA, NA, NA, "Sdl", NA, NA),
  eststage1 = c(NA, NA, NA, NA, "Sdl", NA, NA),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, NA, 0.345, 0.054),
  type = c(1, 1, 1, 1, 1, 3, 3), type_t12 = c(1, 2, 1, 2, 1, 1, 1),
  stageframe = lathframeln, historical = TRUE)

# While we do not use MPMs to initialize f_projections3(), we do use MPMs to
# initialize functions start_input() and density_input().
lathmat3ln <- flefko3(year = "all", patch = "all", data = lathvertln,
  stageframe = lathframeln, supplement = lathsupp3, paramnames = mod_params,
  surv_model = surv_model, obs_model = obs_model, size_model = siz_model,
  repst_model = reps_model, fec_model = fec_model, jsurv_model = jsurv_model,
  jobs_model = jobs_model, jsize_model = jsiz_model,
  jrepst_model = jrepst_model, jmatst_model = jmatst_model, reduce = FALSE)

e3m_sv <- start_input(lathmat3ln, stage2 = "Sd", stage1 = "Sd", value = 1000)

dyn7 <- c(TRUE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE,
  FALSE, FALSE, FALSE, FALSE, FALSE)
dst7 <- c(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
dal7 <- c(0.5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
dbe7 <- c(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
```

```
e3d_vr <- density_vr(density_yn = dyn7, style = dst7, alpha = dal7,
  beta = dbe7)

trial7_dvr_1 <- f_projection3(format = 1, data = lathvertln, supplement = lathsupp3,
  paramnames = mod_params, stageframe = lathframeln, nreps = 2,
  surv_model = surv_model, obs_model = obs_model, size_model = siz_model,
  repst_model = reps_model, fec_model = fec_model, jsurv_model = jsurv_model,
  jobs_model = jobs_model, jsize_model = jsiz_model,
  jrepst_model = jrepst_model, jmatst_model = jmatst_model,
  times = 100, stochastic = TRUE, standardize = FALSE, growthonly = TRUE,
  integeronly = FALSE, substoch = 0, sp_density = 0, start_frame = e3m_sv,
  density_vr = e3d_vr)
```

`diff_lM` *Calculate Difference Matrices Between lefkoMat Objects of Equal Dimensions*

Description

Function `diff_lM()` takes two `lefkoMat` objects with completely equal dimensions, including both the size and number of matrices, and gives the matrix differences between each corresponding set.

Usage

```
diff_lM(mpm1, mpm2)
```

Arguments

`mpm1`	The first `lefkoMat` object.
`mpm2`	The second `lefkoMat` object.

Value

An object of class `lefkoDiff`, which is a set of `A`, `U`, and `F` matrices corresponding to the differences between each set of matrices, followed by the `hstages`, `ahstages`, and `labels` elements from each input `lefkoMat` object. Elements labelled with a `1` at the end refer to `mpm1`, while those labelled `2` at the end refer to `mpm2`.

Notes

The exact difference is calculated as the respective matrix in `mpm1` minus the corresponding matrix in `mpm2`.

This function first checks to see if the number of matrices is the same, and then whether the matrix dimensions are the same. If the two sets differ in at least one of these characteristics, then the function will yield a fatal error.

If the lengths and dimensions of the input `lefkoMat` objects are the same, then this will check if the `labels` element is essentially the same. If not, then the function will yield a warning, but will still operate.

Examples

```
sizevector <- c(0, 0, 0, 0, 0, 0, 1, 3, 6, 11, 19.5)
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, 1.5, 1.5, 3.5, 5)
comments <- c("Dormant seed", "1st yr protocorm", "2nd yr protocorm",
  "3rd yr protocorm", "Seedling", "Dormant adult",
  "Extra small adult (1 shoot)", "Small adult (2-4 shoots)",
  "Medium adult (5-7 shoots)", "Large adult (8-14 shoots)",
  "Extra large adult (>14 shoots)")
cypframe_raw <- sf_create(sizes = sizevector, stagenames = stagevector,
  repstatus = repvector, obsstatus = obsvector, matstatus = matvector,
  propstatus = propvector, immstatus = immvector, indataset = indataset,
  binhalfwidth = binvec, comments = comments)

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)

seeds_per_pod <- 5000

cypsupp2_raw <- supplemental(stage3 = c("SD", "P1", "P2", "P3", "SL", "SL", "D",
    "XSm", "SD", "P1"),
  stage2 = c("SD", "SD", "P1", "P2", "P3", "SL", "SL", "SL", "rep", "rep"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, "D", "XSm", NA, NA),
  eststage2 = c(NA, NA, NA, NA, NA, NA, "XSm", "XSm", NA, NA),
  givenrate = c(0.03, 0.15, 0.1, 0.1, 0.1, 0.05, NA, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, NA, NA, NA, NA, (0.5 * seeds_per_pod),
    (0.5 * seeds_per_pod)),
  type =c(1, 1, 1, 1, 1, 1, 1, 1, 3, 3),
  stageframe = cypframe_raw, historical = FALSE)
cypsupp3_raw <- supplemental(stage3 = c("SD", "SD", "P1", "P1", "P2", "P3",
    "SL", "SL", "SL", "D", "D", "SD", "P1"),
  stage2 = c("SD", "SD", "SD", "SD", "P1", "P2", "P3", "SL", "SL", "SL", "SL",
    "rep", "rep"),
  stage1 = c("SD", "rep", "SD", "rep", "SD", "P1", "P2", "P3", "SL", "P3",
    "SL", "mat", "mat"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, "XSm", "D", NA, NA),
  eststage2 = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, "XSm", "XSm", NA, NA),
```

```
  eststage1 = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, "XSm", "XSm", NA, NA),
  givenrate = c(0.01, 0.05, 0.10, 0.20, 0.1, 0.1, 0.05, 0.05, 0.05, NA, NA,
    NA, NA),
  multiplier = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
    (0.5 * seeds_per_pod), (0.5 * seeds_per_pod)),
  type = c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3),
  type_t12 = c(1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1),
  stageframe = cypframe_raw, historical = TRUE)

cypmatrix2rp <- rlefko2(data = cypraw_v1, stageframe = cypframe_raw,
  year = "all", patch = "all", stages = c("stage3", "stage2"),
  size = c("size3added", "size2added"), supplement = cypsupp2_raw,
  yearcol = "year2", patchcol = "patchid", indivcol = "individ")

cypmatrix2r <- rlefko2(data = cypraw_v1, stageframe = cypframe_raw,
  year = "all", stages = c("stage3", "stage2"),
  size = c("size3added", "size2added"), supplement = cypsupp2_raw,
  yearcol = "year2", patchcol = "patchid", indivcol = "individ")

cypmatrix3rp <- rlefko3(data = cypraw_v1, stageframe = cypframe_raw,
  year = "all", patch = "all", stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added", "size1added"), supplement = cypsupp3_raw,
  yearcol = "year2", patchcol = "patchid", indivcol = "individ")

cypmatrix3r <- rlefko3(data = cypraw_v1, stageframe = cypframe_raw,
  year = "all", stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added", "size1added"), supplement = cypsupp3_raw,
  yearcol = "year2", patchcol = "patchid", indivcol = "individ")

cypmatrix2r_3 <- hist_null(cypmatrix2r)
cypmatrix2r_3 <- delete_lM(cypmatrix2r_3, year = 2004)
diff_r <- diff_lM(cypmatrix3r, cypmatrix2r_3)

cypmatrix2rp_3 <- hist_null(cypmatrix2rp)
cypmatrix2rp_3 <- delete_lM(cypmatrix2rp_3, year = 2004)
diff_rp <- diff_lM(cypmatrix3rp, cypmatrix2rp_3)
```

`edit_lM` *Edit lefkoMat or lefkoMatList Object based on Supplemental Data*

Description

Function `edit_lM()` edits existing `lefkoMat` and `lefkoMatList` objects with external data supplied by the user. The effects are similar to function `supplemental()`, though function `edit_lM()` allows individuals matrices within `lefkoMat` objects to be edited after creation, while `supplemental()` provides external data that modifies all matrices within a `lefkoMat` object, or within all the `lefkoMat` objects within a `lefkoMatList` object.

Usage

```
edit_lM(
  mpm,
  pop = NULL,
  patch = NULL,
  year2 = NULL,
  stage3 = NULL,
  stage2 = NULL,
  stage1 = NULL,
  age2 = NULL,
  eststage3 = NULL,
  eststage2 = NULL,
  eststage1 = NULL,
  estage2 = NULL,
  givenrate = NULL,
  offset = NULL,
  multiplier = NULL,
  type = NULL,
  type_t12 = NULL,
  target_mpm = NULL
)
```

Arguments

mpm	The lefkoMat and lefkoMatList object to be edited.
pop	A string vector denoting the populations to be edited. Defaults to NULL, in which case all populations are edited.
patch	A string vector denoting the patches to be edited. Defaults to NULL, in which case all patches are edited.
year2	A string vector denoting the years to be edited. Defaults to NULL, in which case all years are edited.
stage3	The name of the stage in occasion *t*+1 in the transition to be replaced. Abbreviations for groups of stages are also usable (see Notes). Required in all stage-based and age-by-stage MPMs.
stage2	The name of the stage in occasion *t* in the transition to be replaced. Abbreviations for groups of stages are also usable (see Notes). Required in all stage-based and age-by-stage MPMs.
stage1	The name of the stage in occasion *t*-1 in the transition to be replaced. 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.
age2	An integer vector of the ages in occasion *t* to use in transitions to be changed or replaced. Required for all age- and age-by-stage MPMs.
eststage3	The name of the stage 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	The name of the stage 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	The name of the stage 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*.
estage2	The 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 fixed rate or probability to replace for the transition described by stage3, stage2, and stage1.
offset	A numeric vector of fixed numeric values to add to the transitions described by stage3, stage2, stage1, and/or age2.
multiplier	A vector of numeric multipliers for fecundity or for proxy transitions. Defaults to 1.
type	A 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 multiplier. If empty or not provided, then defaults to 1 for survival transition.
type_t12	An optional 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.
target_mpm	If modifying a lefkoMatList object, then this allows the user to specify which MPMs to modify. To modify, enter a vector with the number of each MPM to modify, or enter "all" to modify all MPMs. Defaults to "all".

Value

An edited copy of the original MPM is returned, also as a lefkoMat object.

Notes

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).

See Also

supplemental()

Examples

```
data(cypdata)

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",
  age_offset = 3, NAas0 = TRUE, NRasRep = TRUE)

cyp_rl <- rleslie(data = cypraw_v1, start_age = 0, last_age = 6, continue = TRUE,
  fecage_min = 3, year = "all", pop = NA, patch = "all", yearcol = "year2",
  patchcol = "patchid", indivcol = "individ")

ddd1 <- edit_lM(cyp_rl, age2 = c(0, 1, 2, 3, 4, 5, 6),
  givenrate = c(0.25, 0.25, 0.4, 0.4, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, 2000, 2000, 2000),
  type = c(1, 1, 1, 1, 3, 3, 3))

ddd1 <- edit_lM(ddd1, age2 = 6, multiplier = 1.5, type = 3, patch = "B",
  year2 = "2005")
```

`elasticity3` *Estimate Elasticity of Population Growth Rate to Matrix Elements*

Description

`elasticity3()` is a generic function that returns the elasticity of the population growth rate to the elements of the matrices in a matrix population model. Currently, this function estimates both deterministic and stochastic elasticities, where the growth rate is λ in the former case and the log of the stochastic λ in the latter case. This function is made to handle very large and sparse matrices supplied as `lefkoMat` objects, as lists of matrices, and as individual matrices.

Usage

```
elasticity3(mats, ...)
```

Arguments

`mats`	A lefkoMat object, a population projection matrix, or a list of population projection matrices for which the stable stage distribution is desired.
`...`	Other parameters.

Value

The value returned depends on the class of the `mats` argument.

See Also

elasticity3.lefkoMat()

elasticity3.matrix()

elasticity3.dgCMatrix()

elasticity3.list()

elasticity3.lefkoMatList()

summary.lefkoElas()

Examples

```
# Lathyrus example
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
matvector <- c(0, 0, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)

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

lathvert <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "Volume88", repstracol = "FCODE88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframe, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, censor = TRUE)

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "Sd", "Sdl", "mat"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "rep", "rep", "Sdl"),
  stage1 = c("Sd", "rep", "Sd", "rep", "npr", "npr", "Sd"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, "mat"),
  eststage2 = c(NA, NA, NA, NA, NA, NA, "Sdl"),
  eststage1 = c(NA, NA, NA, NA, NA, NA, "NotAlive"),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, 0.345, 0.054, NA),
  type = c(1, 1, 1, 1, 3, 3, 1), type_t12 = c(1, 2, 1, 2, 1, 1, 1),
  stageframe = lathframe, historical = TRUE)

ehrlen3 <- rlefko3(data = lathvert, stageframe = lathframe, year = "all",
  stages = c("stage3", "stage2", "stage1"), supplement = lathsupp3,
  yearcol = "year2", indivcol = "individ")
```

```
ehrlen3mean <- lmean(ehrlen3)
elasticity3(ehrlen3mean)

# Cypripedium example
data(cypdata)

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 = 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", "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", patch = "all", stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added"), supplement = cypsupp2r,
  yearcol = "year2", patchcol = "patchid", indivcol = "individ")

elasticity3(cypmatrix2r)
```

---

elasticity3.dgCMatrix *Estimate Elasticity of Population Growth Rate of a Single Sparse Matrix*

Description

elasticity3.dgCMatrix() returns the elasticities of lambda to elements of a single matrix. Because this handles only one matrix, the elasticities are inherently deterministic and based on the dominant eigen value as the best metric of the population growth rate.

Usage

```
## S3 method for class 'dgCMatrix'
elasticity3(mats, sparse = "auto", ...)
```

Arguments

mats	An object of class dgCMatrix.
sparse	A text string indicating whether to use sparse matrix encoding ("yes") or dense matrix encoding ("no"). Defaults to "auto", in which case sparse matrix encoding is used with square matrices with at least 50 rows and no more than 50% of elements with values greater than zero.
...	Other parameters.

Value

This function returns a single elasticity matrix in dgCMatrix format.

See Also

elasticity3()

elasticity3.lefkoMat()

elasticity3.list()

elasticity3.matrix()

elasticity3.lefkoMatList()

summary.lefkoElas()

Examples

```
# Lathyrus example
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
matvector <- c(0, 0, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)

lathframe <- sf_create(sizes = sizevector, stagenames = stagevector,
```

```
  repstatus = repvector, obsstatus = obsvector, matstatus = matvector,
  immstatus = immvector, indataset = indataset, binhalfwidth = binvec,
  propstatus = propvector)

lathvert <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "Volume88", repstracol = "FCODE88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframe, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, censor = TRUE)

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "Sd", "Sdl", "mat"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "rep", "rep", "Sdl"),
  stage1 = c("Sd", "rep", "Sd", "rep", "npr", "npr", "Sd"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, "mat"),
  eststage2 = c(NA, NA, NA, NA, NA, NA, "Sdl"),
  eststage1 = c(NA, NA, NA, NA, NA, NA, "NotAlive"),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, 0.345, 0.054, NA),
  type = c(1, 1, 1, 1, 3, 3, 1), type_t12 = c(1, 2, 1, 2, 1, 1, 1),
  stageframe = lathframe, historical = TRUE)

ehrlen3 <- rlefko3(data = lathvert, stageframe = lathframe, year = "all",
  stages = c("stage3", "stage2", "stage1"), supplement = lathsupp3,
  yearcol = "year2", indivcol = "individ")

ehrlen3mean <- lmean(ehrlen3)
elasticity3(ehrlen3mean$A[[1]])
```

`elasticity3.lefkoMat` *Estimate Elasticity of Population Growth Rate of a lefkoMat Object*

Description

`elasticity3.lefkoMat()` returns the elasticities of population growth rate to elements of all `$A` matrices in an object of class `lefkoMat`. If deterministic, then λ is taken as the population growth rate. If stochastic, then stochastic λ, or the stochastic growth rate, is taken as the population growth rate. This function can handle large and sparse matrices, and so can be used with large historical matrices, IPMs, age x stage matrices, as well as smaller ahistorical matrices.

Usage

```
## S3 method for class 'lefkoMat'
elasticity3(
  mats,
  stochastic = FALSE,
  times = 10000,
  tweights = NA,
```

```
  seed = NA,
  sparse = "auto",
  append_mats = FALSE,
  ...
)
```

Arguments

mats	An object of class lefkoMat.
stochastic	A logical value determining whether to conduct a deterministic (FALSE) or stochastic (TRUE) elasticity analysis. Defaults to FALSE.
times	The number of occasions to project forward in stochastic simulation. Defaults to 10,000.
tweights	An optional numeric vector or matrix denoting the probabilities of choosing each matrix in a stochastic projection. If a matrix is input, 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 a vector is input, then the choice of annual matrix is assumed to be independent of the current matrix. Defaults to equal weighting among matrices.
seed	A number to use as a random number seed in stochastic projection.
sparse	A text string indicating whether to use sparse matrix encoding ("yes") or dense matrix encoding ("no"). Defaults to "auto", in which case sparse matrix encoding is used with square matrices with at least 50 rows and no more than 50% of elements with values greater than zero.
append_mats	A logical value indicating whether to include the original A, U, and F matrices in the output lefkoElas object.
...	Other parameters.

Value

This function returns an object of class lefkoElas, which is a list with 8 elements. The first, h_elasmats, is a list of historical elasticity matrices (NULL if an ahMPM is used as input). The second, ah_elasmats, is a list of either ahistorical elasticity matrices if an ahMPM is used as input, or, if an hMPM is used as input, then the result is a list of elasticity matrices in which historical elasticities have been summed by the stage in occasions t and t+1 to produce historically-corrected elasticity matrices, which are equivalent in dimension to ahistorical elasticity matrices but reflect the effects of stage in occasion t-1. The third element, hstages, is a data frame showing historical stage pairs (NULL if ahMPM used as input). The fourth element, agestages, shows age-stage combinations in the order used in age-by-stage MPMs, if suppled. The fifth element, ahstages, is a data frame showing the order of ahistorical stages. The last 3 elements are the A, U, and F portions of the input.

Notes

Deterministic elasticities are estimated as eqn. 9.72 in Caswell (2001, Matrix Population Models). Stochastic elasticities are estimated as eqn. 14.99 in Caswell (2001). Note that stochastic elasticities are of the stochastic λ, while stochastic sensitivities are with regard to the log of the stochastic λ.

Speed can sometimes be increased by shifting from automatic sparse matrix determination to forced dense or sparse matrix projection. This will most likely occur when matrices have between 30 and 300 rows and columns. Defaults work best when matrices are very small and dense, or very large and sparse.

The `time_weights`, `steps`, and `force_sparse` arguments are now deprecated. Instead, please use the `tweights`, `times`, and `sparse` arguments.

See Also

`elasticity3()`

`elasticity3.dgCMatrix()`

`elasticity3.matrix()`

`elasticity3.list()`

`elasticity3.lefkoMatList()`

`summary.lefkoElas()`

Examples

```
# Lathyrus example
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
matvector <- c(0, 0, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)

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

lathvert <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "Volume88", repstracol = "FCODE88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframe, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, censor = TRUE)

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "Sd", "Sdl", "mat"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "rep", "rep", "Sdl"),
  stage1 = c("Sd", "rep", "Sd", "rep", "npr", "npr", "Sd"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, "mat"),
  eststage2 = c(NA, NA, NA, NA, NA, NA, "Sdl"),
  eststage1 = c(NA, NA, NA, NA, NA, NA, "NotAlive"),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA, NA),
```

```
  multiplier = c(NA, NA, NA, NA, 0.345, 0.054, NA),
  type = c(1, 1, 1, 1, 3, 3, 1), type_t12 = c(1, 2, 1, 2, 1, 1, 1),
  stageframe = lathframe, historical = TRUE)

ehrlen3 <- rlefko3(data = lathvert, stageframe = lathframe, year = "all",
  stages = c("stage3", "stage2", "stage1"), supplement = lathsupp3,
  yearcol = "year2", indivcol = "individ")

elasticity3(ehrlen3, stochastic = TRUE)

# Cypripedium example
data(cypdata)

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 = 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", "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", patch = "all", stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added"), supplement = cypsupp2r,
  yearcol = "year2", patchcol = "patchid", indivcol = "individ")

elasticity3(cypmatrix2r)
```

`elasticity3.lefkoMatList` *Estimate Elasticity of Population Growth Rate of a lefkoMatList Object*

Description

`elasticity3.lefkoMatList()` returns the elasticities of population growth growth rate to elements of all `A` matrices in all bootstrapped `lefkoMat` objects within an entered object of class `lefkoMatList`. If deterministic, then λ is taken as the population growth rate. If stochastic, then stochastic λ, or the stochastic growth rate, is taken as the population growth rate. This function can handle large and sparse matrices, and so can be used with large historical matrices, IPMs, age x stage matrices, as well as smaller ahistorical matrices.

Usage

```
## S3 method for class 'lefkoMatList'
elasticity3(
  mats,
  stochastic = FALSE,
  times = 10000,
  tweights = NA,
  seed = NA,
  sparse = "auto",
  append_mats = FALSE,
  ...
)
```

Arguments

`mats`	An object of class `lefkoMatList`.
`stochastic`	A logical value determining whether to conduct a deterministic (FALSE) or stochastic (TRUE) elasticity analysis. Defaults to FALSE.
`times`	The number of occasions to project forward in stochastic simulation. Defaults to 10,000.
`tweights`	An optional numeric vector or matrix denoting the probabilities of choosing each matrix in a stochastic projection. If a matrix is input, 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 a vector is input, then the choice of annual matrix is assumed to be independent of the current matrix. Defaults to equal weighting among matrices.
`seed`	A number to use as a random number seed in stochastic projection.

sparse	A text string indicating whether to use sparse matrix encoding ("yes") or dense matrix encoding ("no"). Defaults to "auto", in which case sparse matrix encoding is used with square matrices with at least 50 rows and no more than 50% of elements with values greater than zero.
append_mats	A logical value indicating whether to include the original A, U, and F matrices in the output lefkoElas object.
...	Other parameters.

Value

This function returns a list with two elements. The first is an object of class lefkoElas that contains the mean elasticity matrices of the bootstrapped sensitivity matrices. The second is a list containing lefkoElas objects giving the elasticity matrices of all bootstrapped matrices.

Within these lists are objects of class lefkoElas, which are comprised of lists of 8 elements. The first, h_elasmats, is a list of historical elasticity matrices (NULL if an ahMPM is used as input). The second, ah_elasmats, is a list of either ahistorical elasticity matrices if an ahMPM is used as input, or, if an hMPM is used as input, then the result is a list of elasticity matrices in which historical elasticities have been summed by the stage in occasions t and t+1 to produce historically-corrected elasticity matrices, which are equivalent in dimension to ahistorical elasticity matrices but reflect the effects of stage in occasion t-1. The third element, hstages, is a data frame showing historical stage pairs (NULL if ahMPM used as input). The fourth element, agestages, shows age-stage combinations in the order used in age-by-stage MPMs, if suppled. The fifth element, ahstages, is a data frame showing the order of ahistorical stages. The last 3 elements are the A, U, and F portions of the input.

Notes

Deterministic elasticities are estimated as eqn. 9.72 in Caswell (2001, Matrix Population Models). Stochastic elasticities are estimated as eqn. 14.99 in Caswell (2001). Note that stochastic elasticities are of the stochastic λ, while stochastic sensitivities are with regard to the log of the stochastic λ.

Speed can sometimes be increased by shifting from automatic sparse matrix determination to forced dense or sparse matrix projection. This will most likely occur when matrices have between 30 and 300 rows and columns. Defaults work best when matrices are very small and dense, or very large and sparse.

The time_weights, steps, and force_sparse arguments are now deprecated. Instead, please use the tweights, times, and sparse arguments.

See Also

elasticity3()

elasticity3.lefkoMat()

elasticity3.dgCMatrix()

elasticity3.matrix()

elasticity3.list()

summary.lefkoElas()

Examples

```
# Lathyrus example
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
matvector <- c(0, 0, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)

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

lathvert <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "Volume88", repstracol = "FCODE88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframe, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, censor = TRUE)

lathvert_boot <- bootstrap3(lathvert, reps = 3)

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "Sd", "Sdl", "mat"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "rep", "rep", "Sdl"),
  stage1 = c("Sd", "rep", "Sd", "rep", "npr", "npr", "Sd"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, "mat"),
  eststage2 = c(NA, NA, NA, NA, NA, NA, "Sdl"),
  eststage1 = c(NA, NA, NA, NA, NA, NA, "NotAlive"),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, 0.345, 0.054, NA),
  type = c(1, 1, 1, 1, 3, 3, 1), type_t12 = c(1, 2, 1, 2, 1, 1, 1),
  stageframe = lathframe, historical = TRUE)

ehrlen3_boot <- rlefko3(data = lathvert_boot, stageframe = lathframe,
  year = "all", stages = c("stage3", "stage2", "stage1"),
  supplement = lathsupp3, yearcol = "year2", indivcol = "individ")

elasticity3(ehrlen3_boot, stochastic = TRUE)

# Cypripedium example
data(cypdata)

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 = 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)

cypraw_v1_boot <- bootstrap3(cypraw_v1, reps = 3)

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_boot <- rlefko2(data = cypraw_v1_boot, stageframe = cypframe_raw,
  year = "all", patch = "all", stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added"), supplement = cypsupp2r,
  yearcol = "year2", patchcol = "patchid", indivcol = "individ")

elasticity3(cypmatrix2r_boot)
```

`elasticity3.list` *Estimate Elasticity of Population Growth Rate of a List of Matrices*

Description

`elasticity3.list()` returns the elasticities of lambda to elements of a single matrix. This function can handle large and sparse matrices, and so can be used with large historical matrices, IPMs, age x stage matrices, as well as smaller ahistorical matrices.

Usage

```
## S3 method for class 'list'
elasticity3(
  mats,
  stochastic = FALSE,
  times = 10000,
  tweights = NA,
  historical = FALSE,
  seed = NA,
  sparse = "auto",
  append_mats = FALSE,
  ...
)
```

Arguments

mats	A list of objects of class matrix or dgCMatrix.
stochastic	A logical value determining whether to conduct a deterministic (FALSE) or stochastic (TRUE) elasticity analysis. Defaults to FALSE.
times	The number of occasions to project forward in stochastic simulation. Defaults to 10,000.
tweights	An optional numeric vector or matrix denoting the probabilities of choosing each matrix in a stochastic projection. If a matrix is input, 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 a vector is input, then the choice of annual matrix is assumed to be independent of the current matrix. Defaults to equal weighting among matrices.
historical	A logical value denoting whether the input matrices are historical. Defaults to FALSE.
seed	A number to use as a random number seed in stochastic projection.
sparse	A text string indicating whether to use sparse matrix encoding ("yes") or dense matrix encoding ("no"). Defaults to "auto", in which case sparse matrix encoding is used with square matrices with at least 50 rows and no more than 50% of elements with values greater than zero.
append_mats	A logical value indicating whether to include the original matrices input as object mats in the output lefkoElas object.
...	Other parameters.

Value

This function returns an object of class lefkoElas, which is a list with 8 elements. The first, h_elasmats, is a list of historical elasticity matrices, though in the standard list case it returns a NULL value. The second, ah_elasmats, is a list of ahistorical elasticity matrices. The third element, hstages, the fourth element, agestages, and the fifth element, ahstages, are set to NULL. The last 3 elements are the original A matrices in element A, followed by NULL values for the U and F elements.

Notes

Deterministic elasticities are estimated as eqn. 9.72 in Caswell (2001, Matrix Population Models). Stochastic elasticities are estimated as eqn. 14.99 in Caswell (2001). Note that stochastic elasticities are of stochastic λ, while stochastic sensitivities are with regard to the log of the stochastic λ.

Speed can sometimes be increased by shifting from automatic sparse matrix determination to forced dense or sparse matrix projection. This will most likely occur when matrices have between 30 and 300 rows and columns. Defaults work best when matrices are very small and dense, or very large and sparse.

The `time_weights`, `steps`, and `force_sparse` arguments are now deprecated. Instead, please use the `tweights`, `times`, and `sparse` arguments.

See Also

`elasticity3()`

`elasticity3.lefkoMat()`

`elasticity3.matrix()`

`elasticity3.dgCMatrix()`

`elasticity3.lefkoMatList()`

`summary.lefkoElas()`

Examples

```
# Lathyrus example
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
matvector <- c(0, 0, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)

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

lathvert <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "Volume88", repstracol = "FCODE88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframe, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, censor = TRUE)

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "Sd", "Sdl", "mat"),
```

```
  stage2 = c("Sd", "Sd", "Sd", "Sd", "rep", "rep", "Sdl"),
  stage1 = c("Sd", "rep", "Sd", "rep", "npr", "npr", "Sd"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, "mat"),
  eststage2 = c(NA, NA, NA, NA, NA, NA, "Sdl"),
  eststage1 = c(NA, NA, NA, NA, NA, NA, "NotAlive"),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, 0.345, 0.054, NA),
  type = c(1, 1, 1, 1, 3, 3, 1), type_t12 = c(1, 2, 1, 2, 1, 1, 1),
  stageframe = lathframe, historical = TRUE)

ehrlen3 <- rlefko3(data = lathvert, stageframe = lathframe, year = "all",
  stages = c("stage3", "stage2", "stage1"), supplement = lathsupp3,
  yearcol = "year2", indivcol = "individ")

elasticity3(ehrlen3$A, stochastic = TRUE)

# Cypripedium example
data(cypdata)

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 = 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", "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", patch = "all", stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added"), supplement = cypsupp2r,
  yearcol = "year2", patchcol = "patchid", indivcol = "individ")

elasticity3(cypmatrix2r$A)
```

`elasticity3.matrix` *Estimate Elasticity of Population Growth Rate of a Single Matrix*

Description

`elasticity3.matrix()` returns the elasticities of lambda to elements of a single matrix. Because this handles only one matrix, the elasticities are inherently deterministic and based on the dominant eigen value as the best metric of the population growth rate. This function can handle large and sparse matrices, and so can be used with large historical matrices, IPMs, age x stage matrices, as well as smaller ahistorical matrices.

Usage

```
## S3 method for class 'matrix'
elasticity3(mats, sparse = "auto", ...)
```

Arguments

`mats`	An object of class `matrix`.
`sparse`	A text string indicating whether to use sparse matrix encoding (`"yes"`) or dense matrix encoding (`"no"`). Defaults to `"auto"`, in which case sparse matrix encoding is used with square matrices with at least 50 rows and no more than 50% of elements with values greater than zero.
`...`	Other parameters.

Value

This function returns a single elasticity matrix.

Notes

Speed can sometimes be increased by shifting from automatic sparse matrix determination to forced dense or sparse matrix projection. This will most likely occur when matrices have between 30 and 300 rows and columns. Defaults work best when matrices are very small and dense, or very large and sparse.

The `force_sparse` argument is now deprecated. Please use `sparse` instead.

See Also

elasticity3()

elasticity3.lefkoMat()

elasticity3.list()

elasticity3.dgCMatrix()

elasticity3.lefkoMatList()

summary.lefkoElas()

Examples

```
# Lathyrus example
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
matvector <- c(0, 0, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)

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

lathvert <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "Volume88", repstracol = "FCODE88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframe, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, censor = TRUE)

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "Sd", "Sdl", "mat"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "rep", "rep", "Sdl"),
  stage1 = c("Sd", "rep", "Sd", "rep", "npr", "npr", "Sd"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, "mat"),
  eststage2 = c(NA, NA, NA, NA, NA, NA, "Sdl"),
  eststage1 = c(NA, NA, NA, NA, NA, NA, "NotAlive"),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, 0.345, 0.054, NA),
  type = c(1, 1, 1, 1, 3, 3, 1), type_t12 = c(1, 2, 1, 2, 1, 1, 1),
  stageframe = lathframe, historical = TRUE)

ehrlen3 <- rlefko3(data = lathvert, stageframe = lathframe, year = "all",
  stages = c("stage3", "stage2", "stage1"), supplement = lathsupp3,
  yearcol = "year2", indivcol = "individ")
```

```
ehrlen3mean <- lmean(ehrlen3)
elasticity3(ehrlen3mean$A[[1]])
```

---

flefko2 *Create Function-based Ahistorical Matrix Projection Model*

Description

Function `flefko2()` returns ahistorical MPMs corresponding to the patches and occasions given, including the associated component transition and fecundity matrices, a data frame detailing the characteristics of the ahistorical stages used, and a data frame characterizing the patch and occasion combinations corresponding to these matrices.

Usage

```
flefko2(
  year = "all",
  patch = "all",
  stageframe,
  supplement = NULL,
  repmatrix = NULL,
  overwrite = NULL,
  data = NULL,
  modelsuite = NULL,
  surv_model = NULL,
  obs_model = NULL,
  size_model = NULL,
  sizeb_model = NULL,
  sizec_model = NULL,
  repst_model = NULL,
  fec_model = NULL,
  jsurv_model = NULL,
  jobs_model = NULL,
  jsize_model = NULL,
  jsizeb_model = NULL,
  jsizec_model = NULL,
  jrepst_model = NULL,
  jmatst_model = NULL,
  paramnames = NULL,
  inda = NULL,
  indb = NULL,
  indc = NULL,
  annua = NULL,
  annub = NULL,
  annuc = NULL,
  surv_dev = 0,
```

```
  obs_dev = 0,
  size_dev = 0,
  sizeb_dev = 0,
  sizec_dev = 0,
  repst_dev = 0,
  fec_dev = 0,
  jsurv_dev = 0,
  jobs_dev = 0,
  jsize_dev = 0,
  jsizeb_dev = 0,
  jsizec_dev = 0,
  jrepst_dev = 0,
  jmatst_dev = 0,
  density = NA,
  fecmod = 1,
  random.inda = FALSE,
  random.indb = FALSE,
  random.indc = FALSE,
  negfec = FALSE,
  ipm_method = "CDF",
  reduce = FALSE,
  simple = FALSE,
  err_check = FALSE,
  exp_tol = 700,
  theta_tol = 1e+08,
  sparse_output = FALSE
)
```

Arguments

year	A variable corresponding to the observation occasion, or a set of such values, given in values associated with the year term used in linear model development. Defaults to "all", in which case matrices will be estimated for all occasions.
patch	A variable designating which patches or subpopulations will have matrices estimated. Defaults to "all", but can also be set to specific patch names or a vector thereof.
stageframe	An object of class stageframe. These objects are generated by function sf_create(), and include information on the size, observation status, propagule status, reproduction status, immaturity status, maturity status, stage group, size bin widths, and other key characteristics of each ahistorical stage.
supplement	An optional data frame of class lefkoSD that provides supplemental data that should be incorporated into the MPM. Three kinds of data may be integrated this way: transitions to be estimated via the use of proxy transitions, transition overwrites from the literature or supplemental studies, and transition multipliers for survival and fecundity. This data frame should be produced using the supplemental() function. Can be used in place of or in addition to an overwrite table (see overwrite below) and a reproduction matrix (see repmatrix below).

repmatrix — An optional reproduction matrix. This matrix is composed mostly of 0s, with non-zero entries acting as element identifiers and multipliers for fecundity (with 1 equaling full fecundity). If left blank, and no supplement is provided, then flefko2() will assume that all stages marked as reproductive produce offspring at 1x that of estimated fecundity, and that offspring production will yield the first stage noted as propagule or immature. Must be the dimensions of an ahistorical matrix.

overwrite — An optional data frame developed with the overwrite() function describing transitions to be overwritten either with given values or with other estimated transitions. Note that this function supplements overwrite data provided in supplement.

data — The historical vertical demographic data frame used to estimate vital rates (class hfvdata), which is required to initialize times and patches properly. Variable names should correspond to the naming conventions in verticalize3() and historicalize3(). Not required if option modelsuite is set to a vrm_input object.

modelsuite — One of three kinds of lists. The first is a lefkoMod object holding the vital rate models and associated metadata. The second is a lefkoModList object, which is a list of lefkoMod objects generally created to conduct a bootstrapped MPM analysis. Alternatively, an object of class vrm_input may be provided. If given, then surv_model, obs_model, size_model, sizeb_model, sizec_model, repst_model, fec_model, jsurv_model, jobs_model, jsize_model, jsizeb_model, jsizec_model, jrepst_model, jmatst_model, and paramnames are not required. One or more of these models should include size or reproductive status in occasion *t*-1. Although this is optional input, it is recommended, and without it all vital rate model inputs (named XX_model) are required.

surv_model — A linear model predicting survival probability. This can be a model of class glm or glmer, and requires a predicted binomial variable under a logit link. Ignored if modelsuite is provided. This model must have been developed in a modeling exercise testing only the impacts of occasion *t*.

obs_model — A linear model predicting sprouting or observation probability. This can be a model of class glm or glmer, and requires a predicted binomial variable under a logit link. Ignored if modelsuite is provided. This model must have been developed in a modeling exercise testing only the impacts of occasion *t*.

size_model — A linear model predicting primary size. This can be a model of class glm, glmer, glmmTMB, zeroinfl, vglm, lm, or lmer. Ignored if modelsuite is provided. This model must have been developed in a modeling exercise testing only the impacts of occasion *t*.

sizeb_model — A linear model predicting secondary size. This can be a model of class glm, glmer, glmmTMB, zeroinfl, vglm, lm, or lmer. Ignored if modelsuite is provided. This model must have been developed in a modeling exercise testing only the impacts of occasion *t*.

sizec_model — A linear model predicting tertiary size. This can be a model of class glm, glmer, glmmTMB, zeroinfl, vglm, lm, or lmer. Ignored if modelsuite is provided. This model must have been developed in a modeling exercise testing only the impacts of occasion *t*.

repst_model — A linear model predicting reproduction probability. This can be a model of class glm or glmer, and requires a predicted binomial variable under a logit link. Ignored if modelsuite is provided. This model must have been developed in a modeling exercise testing only the impacts of occasion *t*.

fec_model — A linear model predicting fecundity. This can be a model of class glm, glmer, glmmTMB, zeroinfl, vglm, lm, or lmer. Ignored if modelsuite is provided. This model must have been developed in a modeling exercise testing only the impacts of occasion *t*.

jsurv_model — A linear model predicting juvenile survival probability. This can be a model of class glm or glmer, and requires a predicted binomial variable under a logit link. Ignored if modelsuite is provided. This model must have been developed in a modeling exercise testing only the impacts of occasion *t*.

jobs_model — A linear model predicting juvenile sprouting or observation probability. This can be a model of class glm or glmer, and requires a predicted binomial variable under a logit link. Ignored if modelsuite is provided. This model must have been developed in a modeling exercise testing only the impacts of occasion *t*.

jsize_model — A linear model predicting juvenile primary size. This can be a model of class glm, glmer, glmmTMB, zeroinfl, vglm, lm, or lmer. Ignored if modelsuite is provided. This model must have been developed in a modeling exercise testing only the impacts of occasion *t*.

jsizeb_model — A linear model predicting juvenile secondary size. This can be a model of class glm, glmer, glmmTMB, zeroinfl, vglm, lm, or lmer. Ignored if modelsuite is provided. This model must have been developed in a modeling exercise testing only the impacts of occasion *t*.

jsizec_model — A linear model predicting juvenile tertiary size. This can be a model of class glm, glmer, glmmTMB, zeroinfl, vglm, lm, or lmer. Ignored if modelsuite is provided. This model must have been developed in a modeling exercise testing only the impacts of occasion *t*.

jrepst_model — A linear model predicting reproduction probability of a mature individual that was immature in time *t*. This can be a model of class glm or glmer, and requires a predicted binomial variable under a logit link. Ignored if modelsuite is provided. This model must have been developed in a modeling exercise testing only the impacts of occasion *t*.

jmatst_model — A linear model predicting maturity probability of an individual that was immature in time *t*. This can be a model of class glm or glmer, and requires a predicted binomial variable under a logit link. Ignored if modelsuite is provided. This model must have been developed in a modeling exercise testing only the impacts of occasion *t*.

paramnames — A data frame with three columns, the first describing all terms used in linear modeling, the second (must be called mainparams) giving the general model terms that will be used in matrix creation, and the third showing the equivalent terms used in modeling (must be named modelparams). Function create_pm() can be used to create a skeleton paramnames object, which can then be edited. Only required if modelsuite is not supplied.

inda — Can be a single value to use for individual covariate a in all matrices, a pair of values to use for times *t* and *t*-1 in historical matrices, or a vector of such values corresponding to each occasion in the dataset. Defaults to NULL.

indb — Can be a single value to use for individual covariate b in all matrices, a pair of values to use for times *t* and *t*-1 in historical matrices, or a vector of such values corresponding to each occasion in the dataset. Defaults to NULL.

indc — Can be a single value to use for individual covariate c in all matrices, a pair of values to use for times *t* and *t*-1 in historical matrices, or a vector of such values corresponding to each occasion in the dataset. Defaults to NULL.

annua — Can be a single value to use for annual covariate a in all matrices, a pair of values to use for times *t* and *t*-1 in historical matrices, or a vector of such values corresponding to each occasion in the dataset. Defaults to NULL.

annub — Can be a single value to use for annual covariate b in all matrices, a pair of values to use for times *t* and *t*-1 in historical matrices, or a vector of such values corresponding to each occasion in the dataset. Defaults to NULL.

annuc — Can be a single value to use for annual covariate c in all matrices, a pair of values to use for times *t* and *t*-1 in historical matrices, or a vector of such values corresponding to each occasion in the dataset. Defaults to NULL.

surv_dev — A numeric value to be added to the y-intercept in the linear model for survival probability. Defaults to 0.

obs_dev — A numeric value to be added to the y-intercept in the linear model for observation probability. Defaults to 0.

size_dev — A numeric value to be added to the y-intercept in the linear model for primary size. Defaults to 0.

sizeb_dev — A numeric value to be added to the y-intercept in the linear model for secondary size. Defaults to 0.

sizec_dev — A numeric value to be added to the y-intercept in the linear model for tertiary size. Defaults to 0.

repst_dev — A numeric value to be added to the y-intercept in the linear model for probability of reproduction. Defaults to 0.

fec_dev — A numeric value to be added to the y-intercept in the linear model for fecundity. Defaults to 0.

jsurv_dev — A numeric value to be added to the y-intercept in the linear model for juvenile survival probability. Defaults to 0.

jobs_dev — A numeric value to be added to the y-intercept in the linear model for juvenile observation probability. Defaults to 0.

jsize_dev — A numeric value to be added to the y-intercept in the linear model for juvenile primary size. Defaults to 0.

jsizeb_dev — A numeric value to be added to the y-intercept in the linear model for juvenile secondary size. Defaults to 0.

jsizec_dev — A numeric value to be added to the y-intercept in the linear model for juvenile tertiary size. Defaults to 0.

jrepst_dev — A numeric value to be added to the y-intercept in the linear model for juvenile reproduction probability. Defaults to 0.

jmatst_dev — A numeric value to be added to the y-intercept in the linear model for juvenile maturity probability. Defaults to 0.

density — A numeric value indicating density value to use to propagate matrices. Only needed if density is an explanatory term used in one or more vital rate models. Defaults to NA.

fecmod — A scalar multiplier of fecundity. Defaults to 1.0.

random.inda — A logical value denoting whether to treat individual covariate a as a random, categorical variable. Otherwise is treated as a fixed, numeric variable. Defaults to FALSE.

random.indb — A logical value denoting whether to treat individual covariate b as a random, categorical variable. Otherwise is treated as a fixed, numeric variable. Defaults to FALSE.

random.indc — A logical value denoting whether to treat individual covariate c as a random, categorical variable. Otherwise is treated as a fixed, numeric variable. Defaults to FALSE.

negfec — A logical value denoting whether fecundity values estimated to be negative should be reset to 0. Defaults to FALSE.

ipm_method — A string indicating what method to use to estimate size transition probabilities, if size is treated as continuous. Options include: "midpoint", which utilizes the midpoint method; and "CDF", which uses the cumulative distribution function. Defaults to "CDF".

reduce — A logical value denoting whether to remove ahistorical stages associated solely with 0 transitions. These are only removed in cases where the associated row and column sums in ALL matrices estimated equal 0. Defaults to FALSE.

simple — A logical value indicating whether to produce A, U, and F matrices, or only the latter two. Defaults to FALSE, in which case all three are output.

err_check — A logical value indicating whether to append extra information used in matrix calculation within the output list. 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.

sparse_output — A logical value indicating whether to output matrices in sparse format. Defaults to FALSE, in which case all matrices are output in standard matrix format.

Value

If the user inputs a standard lefkoMod or vrm_input object in argument modelsuite, or individual vital rate models are input separately,then this function will return an object of class lefkoMat. If the user inputs an object of class lefkoModList in argument modelsuite, then the output will be an object of class lefkoMatList, in which each element is an object of class lefkoMat.

A lefkoMat object is a list that holds one full matrix projection model and all of its metadata. The structure has the following elements:

A — A list of full projection matrices in order of sorted patches and occasion times. All matrices output in R's matrix class, or in the dgCMatrix class from the Matrix package if sparse.

U	A list of survival transition matrices sorted as in A. All matrices output in R's matrix class, or in the dgCMatrix class from the Matrix package if sparse.
F	A list of fecundity matrices sorted as in A. All matrices output in R's matrix class, or in the dgCMatrix class from the Matrix package if sparse.
hstages	A data frame matrix showing the pairing of ahistorical stages used to create historical stage pairs. Set to NA for ahistorical matrices.
agestages	A data frame showing age-stage pairs. In this function, it is set to NA. Only used in output to function aflefko2().
ahstages	A data frame detailing the characteristics of associated ahistorical stages, in the form of a modified stageframe that includes status as an entry stage through reproduction.
labels	A data frame giving the population, patch, and year of each matrix in order. In flefko2(), only one population may be analyzed at once.
dataqc	A vector showing the numbers of individuals and rows in the vertical dataset used as input.
matrixqc	A short vector describing the number of non-zero elements in U and F matrices, and the number of annual matrices.
modelqc	This is the qc portion of the modelsuite input.
prob_out	An optional element only added if err_check = TRUE. This is a list of vital rate probability matrices, with 7 columns in the order of survival, observation probability, reproduction probability, primary size transition probability, secondary size transition probability, tertiary size transition probability, and probability of juvenile transition to maturity.
allstages	An optional element only added if err_check = TRUE. This is a data frame giving the values used to determine each matrix element capable of being estimated.

Notes

Unlike rlefko2(), rlefko3(), arlefko2(), and rleslie(), this function does not currently distinguish populations. Users wishing to use the same vital rate models across populations should label them as patches (though we do not advise this approach, as populations should typically be treated as statistically independent).

This function will yield incorrect estimates if the models utilized incorporate state in occasion t-1. Only use models developed testing for ahistorical effects.

The default behavior of this function is to estimate fecundity with regards to transitions specified via associated fecundity multipliers in the supplement. If this field is left empty, then fecundity will be estimated at full for all transitions leading from reproductive stages to immature and propagule stages.

Users may at times wish to estimate MPMs using a dataset incorporating multiple patches or subpopulations, but without discriminating between those patches or subpopulations. Should the aim of analysis be a general MPM that does not distinguish these patches or subpopulations, the modelsearch() run should not include patch terms.

Input options including multiple variable names must be entered in the order of variables in occasion t+1 and t. Rearranging the order will lead to erroneous calculations, and may lead to fatal errors.

Care should be taken to match the random status of year and patch to the states of those variables within the `modelsuite`. If they do not match, then they will be treated as zeroes in vital rate estimation.

The `ipm_method` function gives the option of using two different means of estimating the probability of size transition. The midpoint method (`"midpoint"`) refers to the method in which the probability is estimated by first estimating the probability associated with transition from the exact size at the midpoint of the size class using the corresponding probability density function, and then multiplying that value by the bin width of the size class. Doak et al. 2021 (Ecological Monographs) noted that this method can produce biased results, with total size transitions associated with a specific size not totaling to 1.0 and even specific size transition probabilities capable of being estimated at values greater than 1.0. The alternative and default method, `"CDF"`, uses the corresponding cumulative density function to estimate the probability of size transition as the cumulative probability of size transition at the greater limit of the size class minus the cumulative probability of size transition at the lower limit of the size class. The latter method avoids this bias. Note, however, that both methods are exact and unbiased for negative binomial and Poisson distributions.

Under the Gaussian and gamma size distributions, the number of estimated parameters may differ between the two `ipm_method` settings. Because the midpoint method has a tendency to incorporate upward bias in the estimation of size transition probabilities, it is more likely to yield non- zero values when the true probability is extremely close to 0. This will result in the `summary.lefkoMat` function yielding higher numbers of estimated parameters than the `ipm_method = "CDF"` yields in some cases.

Using the `err_check` option will produce a matrix of 7 columns, each characterizing a different vital rate. The product of each row yields an element in the associated `U` matrix. The number and order of elements in each column of this matrix matches the associated matrix in column vector format. Use of this option is generally for the purposes of debugging code.

Individual covariates are treated as categorical only if they are set as random terms. Fixed categorical individual covariates are currently not allowed. However, such terms may be supplied if the `modelsuite` option is set to a `vrm_input` object. In that case, the user should also set the logical random switch for the individual covariate to be used to `TRUE` (e.g., `random.inda = TRUE`).

See Also

`mpm_create()`

`flefko3()`

`aflefko2()`

`arlefko2()`

`fleslie()`

`rlefko3()`

`rlefko2()`

`rleslie()`

Examples

```
# Lathyrus example
data(lathyrus)
```

```
sizevector <- c(0, 4.6, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 1, 2, 3, 4, 5, 6, 7, 8,
  9)
stagevector <- c("Sd", "Sdl", "Dorm", "Sz1nr", "Sz2nr", "Sz3nr", "Sz4nr",
  "Sz5nr", "Sz6nr", "Sz7nr", "Sz8nr", "Sz9nr", "Sz1r", "Sz2r", "Sz3r",
  "Sz4r", "Sz5r", "Sz6r", "Sz7r", "Sz8r", "Sz9r")
repvector <- c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1)
obsvector <- c(0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
matvector <- c(0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0)
indataset <- c(0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 4.6, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
  0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5)

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

lathvertln <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "lnVol88", repstracol = "Intactseed88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframeln,
  stagesize = "sizea", censorcol = "Missing1988", censorkeep = NA,
  NAas0 = TRUE, censor = TRUE)

lathvertln$feca2 <- round(lathvertln$feca2)
lathvertln$feca1 <- round(lathvertln$feca1)
lathvertln$feca3 <- round(lathvertln$feca3)

lathvertln_adults <- subset(lathvertln, stage2index > 2)
surv_model <- glm(alive3 ~ sizea2 + as.factor(patchid),
  data = lathvertln_adults, family = "binomial")

obs_data <- subset(lathvertln_adults, alive3 == 1)
obs_model <- glm(obsstatus3 ~ as.factor(patchid), data = obs_data,
  family = "binomial")

size_data <- subset(obs_data, obsstatus3 == 1)
siz_model <- lm(sizea3 ~ sizea2 + repstatus2 + as.factor(patchid) +
  as.factor(year2), data = size_data)

reps_model <- glm(repstatus3 ~ sizea2 + as.factor(patchid) + as.factor(year2),
  data = size_data, family = "binomial")

fec_data <- subset(lathvertln_adults, repstatus2 == 1)
fec_model <- glm(feca2 ~ sizea2 + as.factor(patchid) + as.factor(year2),
  data = fec_data, family = "poisson")

lathvertln_juvs <- subset(lathvertln, stage2index < 3)
jsurv_model <- glm(alive3 ~ as.factor(patchid), data = lathvertln_juvs,
```

```
  family = "binomial")

jobs_data <- subset(lathvertln_juvs, alive3 == 1)
jobs_model <- glm(obsstatus3 ~ 1, family = "binomial", data = jobs_data)

jsize_data <- subset(jobs_data, obsstatus3 == 1)
jsiz_model <- lm(sizea3 ~ as.factor(year2), data = jsize_data)

jrepst_model <- 0
jmatst_model <- 1

mod_params <- create_pm(name_terms = TRUE)
mod_params$modelparams[3] <- "patchid"
mod_params$modelparams[5] <- "obsstatus3"
mod_params$modelparams[6] <- "sizea3"
mod_params$modelparams[9] <- "repstatus3"
mod_params$modelparams[11] <- "feca2"
mod_params$modelparams[12] <- "sizea2"
mod_params$modelparams[18] <- "repstatus2"

lathsupp2 <- supplemental(stage3 = c("Sd", "Sdl", "Sd", "Sdl"),
  stage2 = c("Sd", "Sd", "rep", "rep"),
  givenrate = c(0.345, 0.054, NA, NA),
  multiplier = c(NA, NA, 0.345, 0.054),
  type = c(1, 1, 3, 3), stageframe = lathframeln, historical = FALSE)

lathmat2ln <- flefko2(year = "all", patch = "all", data = lathvertln,
  stageframe = lathframeln, supplement = lathsupp2, paramnames = mod_params,
  surv_model = surv_model, obs_model = obs_model, size_model = siz_model,
  repst_model = reps_model, fec_model = fec_model, jsurv_model = jsurv_model,
  jobs_model = jobs_model, jsize_model = jsiz_model,
  jrepst_model = jrepst_model, jmatst_model = jmatst_model, reduce = FALSE)


# Cypripedium example using three size metrics for classification
data(cypdata)

sizevector_f <- c(0, 0, 0, 0, 0, 0, seq(1, 12, by = 1), seq(0, 9, by = 1),
  seq(0, 8, by = 1), seq(0, 7, by = 1), seq(0, 6, by = 1), seq(0, 5, by = 1),
  seq(0, 4, by = 1), seq(0, 3, by = 1), 0, 1, 2, 0, 1, 0,
  0, 0, 1, 0)
sizebvector_f <- c(0, 0, 0, 0, 0, 0, rep(0, 12), rep(1, 10), rep(2, 9),
  rep(3, 8), rep(4, 7), rep(5, 6), rep(6, 5), rep(7, 4), rep(8, 3), 9, 9, 10,
  0, 1, 1, 2)
sizecvector_f <- c(0, 0, 0, 0, 0, 0, rep(0, 12), rep(0, 10), rep(0, 9),
  rep(0, 8), rep(0, 7), rep(0, 6), rep(0, 5), rep(0, 4), 0, 0, 0, 0, 0, 0,
  1, 1, 1, 1)
stagevector_f <- c("DS", "P1", "P2", "P3", "Sdl", "Dorm", "V1 I0 D0",
  "V2 I0 D0", "V3 I0 D0", "V4 I0 D0", "V5 I0 D0", "V6 I0 D0", "V7 I0 D0",
  "V8 I0 D0", "V9 I0 D0", "V10 I0 D0", "V11 I0 D0", "V12 I0 D0", "V0 I1 D0",
  "V1 I1 D0", "V2 I1 D0", "V3 I1 D0", "V4 I1 D0", "V5 I1 D0", "V6 I1 D0",
  "V7 I1 D0", "V8 I1 D0", "V9 I1 D0", "V0 I2 D0", "V1 I2 D0", "V2 I2 D0",
  "V3 I2 D0", "V4 I2 D0", "V5 I2 D0", "V6 I2 D0", "V7 I2 D0", "V8 I2 D0",
```

```
  "V0 I3 D0", "V1 I3 D0", "V2 I3 D0", "V3 I3 D0", "V4 I3 D0", "V5 I3 D0",
  "V6 I3 D0", "V7 I3 D0", "V0 I4 D0", "V1 I4 D0", "V2 I4 D0", "V3 I4 D0",
  "V4 I4 D0", "V5 I4 D0", "V6 I4 D0", "V0 I5 D0", "V1 I5 D0", "V2 I5 D0",
  "V3 I5 D0", "V4 I5 D0", "V5 I5 D0", "V0 I6 D0", "V1 I6 D0", "V2 I6 D0",
  "V3 I6 D0", "V4 I6 D0", "V0 I7 D0", "V1 I7 D0", "V2 I7 D0", "V3 I7 D0",
  "V0 I8 D0", "V1 I8 D0", "V2 I8 D0", "V0 I9 D0", "V1 I9 D0", "V0 I10 D0",
  "V0 I0 D1", "V0 I1 D1", "V1 I1 D1", "V0 I2 D1")
repvector_f <- c(0, 0, 0, 0, 0, rep(0, 13), rep(1, 59))
obsvector_f <- c(0, 0, 0, 0, 0, 0, rep(1, 71))
matvector_f <- c(0, 0, 0, 0, 0, rep(1, 72))
immvector_f <- c(0, 1, 1, 1, 1, rep(0, 72))
propvector_f <- c(1, rep(0, 76))
indataset_f <- c(0, 0, 0, 0, 0, rep(1, 72))
binvec_f <- c(0, 0, 0, 0, 0, rep(0.5, 72))
binbvec_f <- c(0, 0, 0, 0, 0, rep(0.5, 72))
bincvec_f <- c(0, 0, 0, 0, 0, rep(0.5, 72))

vertframe_f <- sf_create(sizes = sizevector_f, sizesb = sizebvector_f,
  sizesc = sizecvector_f, stagenames = stagevector_f, repstatus = repvector_f,
  obsstatus = obsvector_f, propstatus = propvector_f, immstatus = immvector_f,
  matstatus = matvector_f, indataset = indataset_f, binhalfwidth = binvec_f,
  binhalfwidthb = binbvec_f, binhalfwidthc = bincvec_f)

vert_data_f <- verticalize3(cypdata, noyears = 6, firstyear = 2004,
  individcol = "plantid", blocksize = 4, sizeacol = "Veg.04",
  sizebcol = "Inf.04", sizeccol = "Inf2.04", repstracol = "Inf.04",
  repstrbcol = "Inf2.04", fecacol = "Pod.04", censorcol = "censor",
  censorkeep = 1, censorRepeat = FALSE, stageassign = vertframe_f,
  stagesize = "sizeabc", NAas0 = TRUE, censor = FALSE)

surv_model <- glm(alive3 ~ sizea2 + sizeb2, data = vert_data_f,
  family = "binomial")

obs_data <- subset(vert_data_f, alive3 == 1)
obs_model <- glm(obsstatus3 ~ sizeb2 + as.factor(year2), data = obs_data,
  family = "binomial")

size_data <- subset(obs_data, obsstatus3 == 1)
siz_model <- MASS::glm.nb(sizea3 ~ sizea2 + sizeb2 + as.factor(year2),
  data = size_data)
sizb_model <- glm(sizeb3 ~ sizea2 + sizeb2 + repstatus2 + as.factor(year2),
  data = size_data, family = "poisson")
sizc_model <- glm(sizec3 ~ repstatus2, data = size_data, family = "poisson")

reps_model <- glm(repstatus3 ~ sizea2 + sizeb2 + repstatus2 + as.factor(year2),
  data = size_data, family = "binomial")

fec_data <- subset(vert_data_f, repstatus2 == 1)
fec_model <- glm(feca2 ~ sizeb2 + as.factor(year2), data = fec_data,
  family = "poisson")

mod_params <- create_pm(name_terms = TRUE)
mod_params$modelparams[3] <- "patchid"
```

```
mod_params$modelparams[4] <- "alive3"
mod_params$modelparams[5] <- "obsstatus3"
mod_params$modelparams[6] <- "sizea3"
mod_params$modelparams[9] <- "repstatus3"
mod_params$modelparams[11] <- "feca2"
mod_params$modelparams[12] <- "sizea2"
mod_params$modelparams[18] <- "repstatus2"

vertsupp2f <- supplemental(stage3 = c("DS", "P1", "P2", "P3", "Sdl", "Sdl",
    "Dorm", "V1 I0 D0", "V2 I0 D0", "V3 I0 D0", "DS", "P1"),
  stage2 = c("DS", "DS", "P1", "P2", "P3", "Sdl", "Sdl", "Sdl", "Sdl", "Sdl",
    "rep", "rep"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, "Dorm", "V1 I0 D0", "V2 I0 D0",
    "V3 I0 D0", NA, NA),
  eststage2 = c(NA, NA, NA, NA, NA, NA, "V1 I0 D0", "V1 I0 D0", "V1 I0 D0",
    "V1 I0 D0", NA, NA),
  givenrate = c(0.10, 0.20, 0.20, 0.20, 0.25, 0.40, NA, NA, NA, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, 0.5 * 5000,
    0.5 * 5000),
  type =c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3), stageframe = vertframe_f,
  historical = FALSE)

vert_mats_f2 <- flefko2(stageframe = vertframe_f, supplement = vertsupp2f,
  data = vert_data_f, surv_model = surv_model, obs_model = obs_model,
  size_model = siz_model, sizeb_model = sizb_model, sizec_model = sizc_model,
  repst_model = reps_model, fec_model = fec_model, paramnames = mod_params)
```

`flefko3` *Create Function-based Historical Matrix Projection Model*

Description

Function `flefko3()` returns function-based historical MPMs corresponding to the patches and occasions given, including the associated component transition and fecundity matrices, data frames detailing the characteristics of the ahistorical stages used and historical stage pairs created, and a data frame characterizing the patch and occasion combinations corresponding to these matrices.

Usage

```
flefko3(
  year = "all",
  patch = "all",
  stageframe,
  supplement = NULL,
  repmatrix = NULL,
  overwrite = NULL,
  data = NULL,
  modelsuite = NULL,
```

```
surv_model = NULL,
obs_model = NULL,
size_model = NULL,
sizeb_model = NULL,
sizec_model = NULL,
repst_model = NULL,
fec_model = NULL,
jsurv_model = NULL,
jobs_model = NULL,
jsize_model = NULL,
jsizeb_model = NULL,
jsizec_model = NULL,
jrepst_model = NULL,
jmatst_model = NULL,
paramnames = NULL,
inda = NULL,
indb = NULL,
indc = NULL,
annua = NULL,
annub = NULL,
annuc = NULL,
surv_dev = 0,
obs_dev = 0,
size_dev = 0,
sizeb_dev = 0,
sizec_dev = 0,
repst_dev = 0,
fec_dev = 0,
jsurv_dev = 0,
jobs_dev = 0,
jsize_dev = 0,
jsizeb_dev = 0,
jsizec_dev = 0,
jrepst_dev = 0,
jmatst_dev = 0,
density = NA,
fecmod = 1,
random.inda = FALSE,
random.indb = FALSE,
random.indc = FALSE,
negfec = FALSE,
format = "ehrlen",
ipm_method = "CDF",
reduce = FALSE,
simple = FALSE,
err_check = FALSE,
exp_tol = 700,
theta_tol = 1e+08,
```

```
  sparse_output = FALSE
)
```

Arguments

year	A variable corresponding to the observation occasion, or a set of such values, given in values associated with the year term used in linear model development. Defaults to "all", in which case matrices will be estimated for all occasions.
patch	A variable designating which patches or subpopulations will have matrices estimated. Defaults to "all", but can also be set to specific patch names or a vector thereof.
stageframe	An object of class stageframe. These objects are generated by function sf_create(), and include information on the size, observation status, propagule status, reproduction status, immaturity status, maturity status, stage group, size bin widths, and other key characteristics of each ahistorical stage.
supplement	An optional data frame of class lefkoSD that provides supplemental data that should be incorporated into the MPM. Three kinds of data may be integrated this way: transitions to be estimated via the use of proxy transitions, transition overwrites from the literature or supplemental studies, and transition multipliers for survival and fecundity. This data frame should be produced using the supplemental() function. Can be used in place of or in addition to an overwrite table (see overwrite below) and a reproduction matrix (see repmatrix below).
repmatrix	An optional reproduction matrix. This matrix is composed mostly of 0s, with non-zero entries acting as element identifiers and multipliers for fecundity (with 1 equaling full fecundity). If left blank, and no supplement is provided, then flefko3() will assume that all stages marked as reproductive produce offspring at 1x that of estimated fecundity, and that offspring production will yield the first stage noted as propagule or immature. May be the dimensions of either a historical or an ahistorical matrix. If the latter, then all stages will be used in occasion *t*-1 for each suggested ahistorical transition.
overwrite	An optional data frame developed with the overwrite() function describing transitions to be overwritten either with given values or with other estimated transitions. Note that this function supplements overwrite data provided in supplement.
data	The historical vertical demographic data frame used to estimate vital rates (class hfvdata), which is required to initialize times and patches properly. Variable names should correspond to the naming conventions in verticalize3() and historicalize3(). Not required if option modelsuite is set to a vrm_input object.
modelsuite	One of three kinds of lists. The first is a lefkoMod object holding the vital rate models and associated metadata. The second is a lefkoModList object, which is a list of lefkoMod objects generally created to conduct a bootstrapped MPM analysis. Alternatively, an object of class vrm_input may be provided. If given, then surv_model, obs_model, size_model, sizeb_model, sizec_model, repst_model, fec_model, jsurv_model, jobs_model, jsize_model, jsizeb_model, jsizec_model, jrepst_model, jmatst_model, and paramnames

are not required. One or more of these models should include size or reproductive status in occasion *t*-1. Although this is optional input, it is recommended, and without it all vital rate model inputs (named `XX_model`) are required.

`surv_model` A linear model predicting survival probability. This can be a model of class `glm` or `glmer`, and requires a predicted binomial variable under a logit link. Ignored if `modelsuite` is provided. This model must have been developed in a modeling exercise testing the impacts of occasions *t* and *t*-1.

`obs_model` A linear model predicting sprouting or observation probability. This can be a model of class `glm` or `glmer`, and requires a predicted binomial variable under a logit link. Ignored if `modelsuite` is provided. This model must have been developed in a modeling exercise testing the impacts of occasions *t* and *t*-1.

`size_model` A linear model predicting primary size. This can be a model of class `glm`, `glmer`, `glmmTMB`, `zeroinfl`, `vglm`, `lm`, or `lmer`. Ignored if `modelsuite` is provided. This model must have been developed in a modeling exercise testing the impacts of occasions *t* and *t*-1.

`sizeb_model` A linear model predicting secondary size. This can be a model of class `glm`, `glmer`, `glmmTMB`, `zeroinfl`, `vglm`, `lm`, or `lmer`. Ignored if `modelsuite` is provided. This model must have been developed in a modeling exercise testing the impacts of occasions *t* and *t*-1.

`sizec_model` A linear model predicting tertiary size. This can be a model of class `glm`, `glmer`, `glmmTMB`, `zeroinfl`, `vglm`, `lm`, or `lmer`. Ignored if `modelsuite` is provided. This model must have been developed in a modeling exercise testing the impacts of occasions *t* and *t*-1.

`repst_model` A linear model predicting reproduction probability. This can be a model of class `glm` or `glmer`, and requires a predicted binomial variable under a logit link. Ignored if `modelsuite` is provided. This model must have been developed in a modeling exercise testing the impacts of occasions *t* and *t*-1.

`fec_model` A linear model predicting fecundity. This can be a model of class `glm`, `glmer`, `glmmTMB`, `zeroinfl`, `vglm`, `lm`, or `lmer`. Ignored if `modelsuite` is provided. This model must have been developed in a modeling exercise testing the impacts of occasions *t* and *t*-1.

`jsurv_model` A linear model predicting juvenile survival probability. This can be a model of class `glm` or `glmer`, and requires a predicted binomial variable under a logit link. Ignored if `modelsuite` is provided. This model must have been developed in a modeling exercise testing the impacts of occasions *t* and *t*-1.

`jobs_model` A linear model predicting juvenile sprouting or observation probability. This can be a model of class `glm` or `glmer`, and requires a predicted binomial variable under a logit link. Ignored if `modelsuite` is provided. This model must have been developed in a modeling exercise testing the impacts of occasions *t* and *t*-1.

`jsize_model` A linear model predicting juvenile primary size. This can be a model of class `glm`, `glmer`, `glmmTMB`, `zeroinfl`, `vglm`, `lm`, or `lmer`. Ignored if `modelsuite` is provided. This model must have been developed in a modeling exercise testing the impacts of occasions *t* and *t*-1.

`jsizeb_model` A linear model predicting juvenile secondary size. This can be a model of class `glm`, `glmer`, `glmmTMB`, `zeroinfl`, `vglm`, `lm`, or `lmer`. Ignored if `modelsuite` is

provided. This model must have been developed in a modeling exercise testing the impacts of occasions *t* and *t*-1.

`jsizec_model` A linear model predicting juvenile tertiary size. This can be a model of class `glm`, `glmer`, `glmmTMB`, `zeroinfl`, `vglm`, `lm`, or `lmer`. Ignored if `modelsuite` is provided. This model must have been developed in a modeling exercise testing the impacts of occasions *t* and *t*-1.

`jrepst_model` A linear model predicting reproduction probability of a mature individual that was immature in time *t*. This can be a model of class `glm` or `glmer`, and requires a predicted binomial variable under a logit link. Ignored if `modelsuite` is provided. This model must have been developed in a modeling exercise testing the impacts of occasions *t* and *t*-1.

`jmatst_model` A linear model predicting maturity probability of an individual that was immature in time *t*. This can be a model of class `glm` or `glmer`, and requires a predicted binomial variable under a logit link. Ignored if `modelsuite` is provided. This model must have been developed in a modeling exercise testing the impacts of occasions *t* and *t*-1.

`paramnames` A data frame with three columns, the first describing all terms used in linear modeling, the second (must be called `mainparams`) giving the general model terms that will be used in matrix creation, and the third showing the equivalent terms used in modeling (must be named `modelparams`). Function `create_pm()` can be used to create a skeleton `paramnames` object, which can then be edited. Only required if `modelsuite` is not supplied.

`inda` Can be a single value to use for individual covariate `a` in all matrices, a pair of values to use for times *t* and *t*-1 in historical matrices, or a vector of such values corresponding to each occasion in the dataset. Defaults to `NULL`.

`indb` Can be a single value to use for individual covariate `b` in all matrices, a pair of values to use for times *t* and *t*-1 in historical matrices, or a vector of such values corresponding to each occasion in the dataset. Defaults to `NULL`.

`indc` Can be a single value to use for individual covariate `c` in all matrices, a pair of values to use for times *t* and *t*-1 in historical matrices, or a vector of such values corresponding to each occasion in the dataset. Defaults to `NULL`.

`annua` Can be a single value to use for annual covariate `a` in all matrices, a pair of values to use for times *t* and *t*-1 in historical matrices, or a vector of such values corresponding to each occasion in the dataset. Defaults to `NULL`.

`annub` Can be a single value to use for annual covariate `b` in all matrices, a pair of values to use for times *t* and *t*-1 in historical matrices, or a vector of such values corresponding to each occasion in the dataset. Defaults to `NULL`.

`annuc` Can be a single value to use for annual covariate `c` in all matrices, a pair of values to use for times *t* and *t*-1 in historical matrices, or a vector of such values corresponding to each occasion in the dataset. Defaults to `NULL`.

`surv_dev` A numeric value to be added to the y-intercept in the linear model for survival probability. Defaults to `0`.

`obs_dev` A numeric value to be added to the y-intercept in the linear model for observation probability. Defaults to `0`.

size_dev — A numeric value to be added to the y-intercept in the linear model for primary size. Defaults to 0.

sizeb_dev — A numeric value to be added to the y-intercept in the linear model for secondary size. Defaults to 0.

sizec_dev — A numeric value to be added to the y-intercept in the linear model for tertiary size. Defaults to 0.

repst_dev — A numeric value to be added to the y-intercept in the linear model for probability of reproduction. Defaults to 0.

fec_dev — A numeric value to be added to the y-intercept in the linear model for fecundity. Defaults to 0.

jsurv_dev — A numeric value to be added to the y-intercept in the linear model for juvenile survival probability. Defaults to 0.

jobs_dev — A numeric value to be added to the y-intercept in the linear model for juvenile observation probability. Defaults to 0.

jsize_dev — A numeric value to be added to the y-intercept in the linear model for juvenile primary size. Defaults to 0.

jsizeb_dev — A numeric value to be added to the y-intercept in the linear model for juvenile secondary size. Defaults to 0.

jsizec_dev — A numeric value to be added to the y-intercept in the linear model for juvenile tertiary size. Defaults to 0.

jrepst_dev — A numeric value to be added to the y-intercept in the linear model for juvenile reproduction probability. Defaults to 0.

jmatst_dev — A numeric value to be added to the y-intercept in the linear model for juvenile maturity probability. Defaults to 0.

density — A numeric value indicating density value to use to propagate matrices. Only needed if density is an explanatory term used in one or more vital rate models. Defaults to NA.

fecmod — A scalar multiplier of fecundity. Defaults to 1.0.

random.inda — A logical value denoting whether to treat individual covariate a as a random, categorical variable. Otherwise is treated as a fixed, numeric variable. Defaults to FALSE.

random.indb — A logical value denoting whether to treat individual covariate b as a random, categorical variable. Otherwise is treated as a fixed, numeric variable. Defaults to FALSE.

random.indc — A logical value denoting whether to treat individual covariate c as a random, categorical variable. Otherwise is treated as a fixed, numeric variable. Defaults to FALSE.

negfec — A logical value denoting whether fecundity values estimated to be negative should be reset to 0. Defaults to FALSE.

format — A string indicating whether to estimate matrices in ehrlen format or deVries format. The latter adds one extra prior stage to account for the prior state of newborns. Defaults to ehrlen format.

ipm_method	A string indicating what method to use to estimate size transition probabilities, if size is treated as continuous. Options include: "midpoint", which utilizes the midpoint method; and "CDF", which uses the cumulative distribution function. Defaults to "CDF".
reduce	A logical value denoting whether to remove historical stages associated solely with 0 transitions. These are only removed in cases where the associated row and column sums in ALL matrices estimated equal 0. Defaults to FALSE.
simple	A logical value indicating whether to produce A, U, and F matrices, or only the latter two. Defaults to FALSE, in which case all three are output.
err_check	A logical value indicating whether to append extra information used in matrix calculation within the output list. 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.
sparse_output	A logical value indicating whether to output matrices in sparse format. Defaults to FALSE, in which case all matrices are output in standard matrix format.

Value

If the user inputs a standard lefkoMod or vrm_input object in argument modelsuite, or individual vital rate models are input separately,then this function will return an object of class lefkoMat. If the user inputs an object of class lefkoModList in argument modelsuite, then the output will be an object of class lefkoMatList, in which each element is an object of class lefkoMat.

A lefkoMat object is a list that holds one full matrix projection model and all of its metadata. The structure has the following elements:

A	A list of full projection matrices in order of sorted patches and occasion times. All matrices output in R's matrix class, or in the dgCMatrix class from the Matrix package if sparse.
U	A list of survival transition matrices sorted as in A. All matrices output in R's matrix class, or in the dgCMatrix class from the Matrix package if sparse.
F	A list of fecundity matrices sorted as in A. All matrices output in R's matrix class, or in the dgCMatrix class from the Matrix package if sparse.
hstages	A data frame matrix showing the pairing of ahistorical stages used to create historical stage pairs.
agestages	A data frame showing age-stage pairs. In this function, it is set to NA. Only used in output to function aflefko2().
ahstages	A data frame detailing the characteristics of associated ahistorical stages, in the form of a modified stageframe that includes status as an entry stage through reproduction.
labels	A data frame giving the population, patch, and year of each matrix in order. In flefko3(), only one population may be analyzed at once.

dataqc	A vector showing the numbers of individuals and rows in the vertical dataset used as input.
matrixqc	A short vector describing the number of non-zero elements in U and F matrices, and the number of annual matrices.
modelqc	This is the qc portion of the modelsuite input.
prob_out	An optional element only added if err_check = TRUE. This is a list of vital rate probability matrices, with 7 columns in the order of survival, observation probability, reproduction probability, primary size transition probability, secondary size transition probability, tertiary size transition probability, and probability of juvenile transition to maturity.
allstages	An optional element only added if err_check = TRUE. This is a data frame giving the values used to determine each matrix element capable of being estimated.

Notes

Unlike rlefko2(), rlefko3(), arlefko2(), and rleslie(), this function does not currently distinguish populations. Users wishing to use the same vital rate models across populations should label them as patches (though we do not advise this approach, as populations should typically be treated as statistically independent).

The default behavior of this function is to estimate fecundity with regards to transitions specified via associated fecundity multipliers in the supplement. If this field is left empty, then fecundity will be estimated at full for all transitions leading from reproductive stages to immature and propagule stages.

Users may at times wish to estimate MPMs using a dataset incorporating multiple patches or subpopulations, but without discriminating between those patches or subpopulations. Should the aim of analysis be a general MPM that does not distinguish these patches or subpopulations, the modelsearch() run should not include patch terms.

Input options including multiple variable names must be entered in the order of variables in occasion *t*+1, *t*, and *t*-1. Rearranging the order will lead to erroneous calculations, and will may lead to fatal errors.

The ipm_method function gives the option of using two different means of estimating the probability of size transition. The midpoint method ("midpoint") refers to the method in which the probability is estimated by first estimating the probability associated with transition from the exact size at the midpoint of the size class using the corresponding probability density function, and then multiplying that value by the bin width of the size class. Doak et al. 2021 (Ecological Monographs) noted that this method can produce biased results, with total size transitions associated with a specific size not totaling to 1.0 and even specific size transition probabilities capable of being estimated at values greater than 1.0. The alternative and default method, "CDF", uses the corresponding cumulative density function to estimate the probability of size transition as the cumulative probability of size transition at the greater limit of the size class minus the cumulative probability of size transition at the lower limit of the size class. This latter method avoids this bias. Note, however, that both methods are exact and unbiased for negative binomial and Poisson distributions.

Under the Gaussian and gamma size distributions, the number of estimated parameters may differ between the two ipm_method settings. Because the midpoint method has a tendency to incorporate upward bias in the estimation of size transition probabilities, it is more likely to yield non- zero values when the true probability is extremely close to 0. This will result in the summary.lefkoMat

function yielding higher numbers of estimated parameters than the `ipm_method = "CDF"` yields in some cases.

Using the `err_check` option will produce a matrix of 7 columns, each characterizing a different vital rate. The product of each row yields an element in the associated `U` matrix. The number and order of elements in each column of this matrix matches the associated matrix in column vector format. Use of this option is generally for the purposes of debugging code. ' Individual covariates are treated as categorical only if they are set as random terms. Fixed categorical individual covariates are currently not allowed. However, such terms may be supplied if the `modelsuite` option is set to a `vrm_input` object. In that case, the user should also set the logical random switch for the individual covariate to be used to `TRUE` (e.g., `random.inda = TRUE`).

See Also

`mpm_create()`

`flefko2()`

`aflefko2()`

`arlefko2()`

`fleslie()`

`rlefko3()`

`rlefko2()`

`rleslie()`

Examples

```
# Lathyrus example
data(lathyrus)

sizevector <- c(0, 4.6, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 1, 2, 3, 4, 5, 6, 7, 8,
  9)
stagevector <- c("Sd", "Sdl", "Dorm", "Sz1nr", "Sz2nr", "Sz3nr", "Sz4nr",
  "Sz5nr", "Sz6nr", "Sz7nr", "Sz8nr", "Sz9nr", "Sz1r", "Sz2r", "Sz3r",
  "Sz4r", "Sz5r", "Sz6r", "Sz7r", "Sz8r", "Sz9r")
repvector <- c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1)
obsvector <- c(0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
matvector <- c(0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0)
indataset <- c(0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 4.6, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
  0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5)

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

lathvertln <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
```

```
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "lnVol88", repstracol = "Intactseed88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframeln, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, NAas0 = TRUE, censor = TRUE)

lathvertln$feca2 <- round(lathvertln$feca2)
lathvertln$feca1 <- round(lathvertln$feca1)
lathvertln$feca3 <- round(lathvertln$feca3)

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "mat", "Sd", "Sdl"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "Sdl", "rep", "rep"),
  stage1 = c("Sd", "rep", "Sd", "rep", "Sd", "mat", "mat"),
  eststage3 = c(NA, NA, NA, NA, "mat", NA, NA),
  eststage2 = c(NA, NA, NA, NA, "Sdl", NA, NA),
  eststage1 = c(NA, NA, NA, NA, "Sdl", NA, NA),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, NA, 0.345, 0.054),
  type = c(1, 1, 1, 1, 1, 3, 3), type_t12 = c(1, 2, 1, 2, 1, 1, 1),
  stageframe = lathframeln, historical = TRUE)

lathvertln_adults <- subset(lathvertln, stage2index > 2)
surv_model <- glm(alive3 ~ sizea2 + sizea1 + as.factor(patchid) +
  as.factor(year2), data = lathvertln_adults, family = "binomial")

obs_data <- subset(lathvertln_adults, alive3 == 1)
obs_model <- glm(obsstatus3 ~ as.factor(patchid), data = obs_data,
  family = "binomial")

size_data <- subset(obs_data, obsstatus3 == 1)
siz_model <- lm(sizea3 ~ sizea2 + sizea1 + repstatus1 + as.factor(patchid) +
  as.factor(year2), data = size_data)

reps_model <- glm(repstatus3 ~ sizea2 + sizea1 + as.factor(patchid) +
  as.factor(year2), data = size_data, family = "binomial")

fec_data <- subset(lathvertln_adults, repstatus2 == 1)
fec_model <- glm(feca2 ~ sizea2 + sizea1 + repstatus1 + as.factor(patchid),
  data = fec_data, family = "poisson")

lathvertln_juvs <- subset(lathvertln, stage2index < 3)
jsurv_model <- glm(alive3 ~ as.factor(patchid), data = lathvertln_juvs,
  family = "binomial")

jobs_data <- subset(lathvertln_juvs, alive3 == 1)
jobs_model <- glm(obsstatus3 ~ 1, family = "binomial", data = jobs_data)

jsize_data <- subset(jobs_data, obsstatus3 == 1)
jsiz_model <- lm(sizea3 ~ as.factor(year2), data = jsize_data)

jrepst_model <- 0
jmatst_model <- 1
```

```
mod_params <- create_pm(name_terms = TRUE)
mod_params$modelparams[3] <- "patchid"
mod_params$modelparams[4] <- "alive3"
mod_params$modelparams[5] <- "obsstatus3"
mod_params$modelparams[6] <- "sizea3"
mod_params$modelparams[9] <- "repstatus3"
mod_params$modelparams[11] <- "feca2"
mod_params$modelparams[12] <- "sizea2"
mod_params$modelparams[13] <- "sizea1"
mod_params$modelparams[18] <- "repstatus2"
mod_params$modelparams[19] <- "repstatus1"

lathmat3ln <- flefko3(year = "all", patch = "all", data = lathvertln,
  stageframe = lathframeln, supplement = lathsupp3, paramnames = mod_params,
  surv_model = surv_model, obs_model = obs_model, size_model = siz_model,
  repst_model = reps_model, fec_model = fec_model, jsurv_model = jsurv_model,
  jobs_model = jobs_model, jsize_model = jsiz_model,
  jrepst_model = jrepst_model, jmatst_model = jmatst_model, reduce = FALSE)


# Cypripedium example using three size metrics for classification
data(cypdata)

sizevector_f <- c(0, 0, 0, 0, 0, 0, seq(1, 12, by = 1), seq(0, 9, by = 1),
  seq(0, 8, by = 1), seq(0, 7, by = 1), seq(0, 6, by = 1), seq(0, 5, by = 1),
  seq(0, 4, by = 1), seq(0, 3, by = 1), 0, 1, 2, 0, 1, 0,
  0, 0, 1, 0)
sizebvector_f <- c(0, 0, 0, 0, 0, 0, rep(0, 12), rep(1, 10), rep(2, 9),
  rep(3, 8), rep(4, 7), rep(5, 6), rep(6, 5), rep(7, 4), rep(8, 3), 9, 9, 10,
  0, 1, 1, 2)
sizecvector_f <- c(0, 0, 0, 0, 0, 0, rep(0, 12), rep(0, 10), rep(0, 9),
  rep(0, 8), rep(0, 7), rep(0, 6), rep(0, 5), rep(0, 4), 0, 0, 0, 0, 0, 0,
  1, 1, 1, 1)
stagevector_f <- c("DS", "P1", "P2", "P3", "Sdl", "Dorm", "V1 I0 D0",
  "V2 I0 D0", "V3 I0 D0", "V4 I0 D0", "V5 I0 D0", "V6 I0 D0", "V7 I0 D0",
  "V8 I0 D0", "V9 I0 D0", "V10 I0 D0", "V11 I0 D0", "V12 I0 D0", "V0 I1 D0",
  "V1 I1 D0", "V2 I1 D0", "V3 I1 D0", "V4 I1 D0", "V5 I1 D0", "V6 I1 D0",
  "V7 I1 D0", "V8 I1 D0", "V9 I1 D0", "V0 I2 D0", "V1 I2 D0", "V2 I2 D0",
  "V3 I2 D0", "V4 I2 D0", "V5 I2 D0", "V6 I2 D0", "V7 I2 D0", "V8 I2 D0",
  "V0 I3 D0", "V1 I3 D0", "V2 I3 D0", "V3 I3 D0", "V4 I3 D0", "V5 I3 D0",
  "V6 I3 D0", "V7 I3 D0", "V0 I4 D0", "V1 I4 D0", "V2 I4 D0", "V3 I4 D0",
  "V4 I4 D0", "V5 I4 D0", "V6 I4 D0", "V0 I5 D0", "V1 I5 D0", "V2 I5 D0",
  "V3 I5 D0", "V4 I5 D0", "V5 I5 D0", "V0 I6 D0", "V1 I6 D0", "V2 I6 D0",
  "V3 I6 D0", "V4 I6 D0", "V0 I7 D0", "V1 I7 D0", "V2 I7 D0", "V3 I7 D0",
  "V0 I8 D0", "V1 I8 D0", "V2 I8 D0", "V0 I9 D0", "V1 I9 D0", "V0 I10 D0",
  "V0 I0 D1", "V0 I1 D1", "V1 I1 D1", "V0 I2 D1")
repvector_f <- c(0, 0, 0, 0, 0, rep(0, 13), rep(1, 59))
obsvector_f <- c(0, 0, 0, 0, 0, 0, rep(1, 71))
matvector_f <- c(0, 0, 0, 0, 0, rep(1, 72))
immvector_f <- c(0, 1, 1, 1, 1, rep(0, 72))
propvector_f <- c(1, rep(0, 76))
indataset_f <- c(0, 0, 0, 0, 0, rep(1, 72))
binvec_f <- c(0, 0, 0, 0, 0, rep(0.5, 72))
```

```
binbvec_f <- c(0, 0, 0, 0, 0, rep(0.5, 72))
bincvec_f <- c(0, 0, 0, 0, 0, rep(0.5, 72))

vertframe_f <- sf_create(sizes = sizevector_f, sizesb = sizebvector_f,
  sizesc = sizecvector_f, stagenames = stagevector_f, repstatus = repvector_f,
  obsstatus = obsvector_f, propstatus = propvector_f, immstatus = immvector_f,
  matstatus = matvector_f, indataset = indataset_f, binhalfwidth = binvec_f,
  binhalfwidthb = binbvec_f, binhalfwidthc = bincvec_f)

vert_data_f <- verticalize3(cypdata, noyears = 6, firstyear = 2004,
  individcol = "plantid", blocksize = 4, sizeacol = "Veg.04",
  sizebcol = "Inf.04", sizeccol = "Inf2.04", repstracol = "Inf.04",
  repstrbcol = "Inf2.04", fecacol = "Pod.04", censorcol = "censor",
  censorkeep = 1, censorRepeat = FALSE, stageassign = vertframe_f,
  stagesize = "sizeabc", NAas0 = TRUE, censor = FALSE)

vertsupp3f <- supplemental(stage3 = c("DS", "P1", "DS", "P1", "P2", "P2", "P3",
    "Sdl", "Sdl", "Sdl", "Dorm", "V1 I0 D0", "V2 I0 D0", "V3 I0 D0", "Dorm",
    "V1 I0 D0", "V2 I0 D0", "V3 I0 D0", "mat", "mat", "mat", "mat", "DS", "P1"),
  stage2 = c("DS", "DS", "DS", "DS", "P1", "P1", "P2", "P3", "Sdl", "Sdl", "Sdl",
    "Sdl", "Sdl", "Sdl", "Sdl", "Sdl", "Sdl", "Sdl", "Dorm", "V1 I0 D0",
    "V2 I0 D0", "V3 I0 D0", "rep", "rep"),
  stage1 = c("DS", "DS", "rep", "rep", "DS", "rep", "P1", "P2", "P3", "Sdl",
    "Sdl", "Sdl", "Sdl", "Sdl", "P3", "P3", "P3", "P3", "Sdl", "Sdl", "Sdl",
    "Sdl", "mat", "mat"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, "Dorm", "V1 I0 D0",
    "V2 I0 D0", "V3 I0 D0", "Dorm", "V1 I0 D0", "V2 I0 D0", "V3 I0 D0", "mat",
    "mat", "mat", "mat", NA, NA),
  eststage2 = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, "V1 I0 D0", "V1 I0 D0",
    "V1 I0 D0", "V1 I0 D0", "V1 I0 D0", "V1 I0 D0", "V1 I0 D0", "V1 I0 D0",
    "Dorm", "V1 I0 D0", "V2 I0 D0", "V3 I0 D0", NA, NA),
  eststage1 = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, "V1 I0 D0", "V1 I0 D0",
    "V1 I0 D0", "V1 I0 D0", "V1 I0 D0", "V1 I0 D0", "V1 I0 D0", "V1 I0 D0",
    "V1 I0 D0", "V1 I0 D0", "V1 I0 D0", "V1 I0 D0", NA, NA),
  givenrate = c(0.10, 0.20, 0.10, 0.20, 0.20, 0.20, 0.20, 0.25, 0.40, 0.40, NA,
    NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
    NA, NA, NA, NA, NA, NA, NA, 0.5 * 5000, 0.5 * 5000),
  type =c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    3, 3),
  type_t12 = c(1, 1, 2, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    1, 1, 1),
  stageframe = vertframe_f, historical = TRUE)

surv_model <- glm(alive3 ~ sizea2 + sizeb2, data = vert_data_f,
  family = "binomial")

obs_data <- subset(vert_data_f, alive3 == 1)
obs_model <- glm(obsstatus3 ~ sizeb2 + sizec1 + as.factor(year2),
  data = obs_data, family = "binomial")

size_data <- subset(obs_data, obsstatus3 == 1)
siz_model <- MASS::glm.nb(sizea3 ~ sizea2 + sizea1 + sizeb1, data = size_data)
```

```
sizb_model <- glm(sizeb3 ~ sizea2 + sizeb2 + sizec1 + repstatus2 + repstatus1 +
  as.factor(year2), data = size_data, family = "poisson")
sizc_model <- glm(sizec3 ~ sizea1 + repstatus2, data = size_data,
  family = "poisson")

reps_model <- glm(repstatus3 ~ sizea2 + sizeb2 + repstatus2 + as.factor(year2),
  data = size_data, family = "binomial")

fec_data <- subset(vert_data_f, repstatus2 == 1)
fec_model <- glm(feca2 ~ sizeb2 + as.factor(year2), data = fec_data,
  family = "poisson")

mod_params <- create_pm(name_terms = TRUE)
mod_params$modelparams[3] <- "patchid"
mod_params$modelparams[4] <- "alive3"
mod_params$modelparams[5] <- "obsstatus3"
mod_params$modelparams[6] <- "sizea3"
mod_params$modelparams[9] <- "repstatus3"
mod_params$modelparams[11] <- "feca2"
mod_params$modelparams[12] <- "sizea2"
mod_params$modelparams[13] <- "sizea1"
mod_params$modelparams[18] <- "repstatus2"
mod_params$modelparams[19] <- "repstatus1"

vert_mats_f3 <- flefko3(stageframe = vertframe_f, supplement = vertsupp3f,
  data = vert_data_f, surv_model = surv_model, obs_model = obs_model,
  size_model = siz_model, sizeb_model = sizb_model, sizec_model = sizc_model,
  repst_model = reps_model, fec_model = fec_model, paramnames = mod_params,
  sparse_output = TRUE)
```

`fleslie` *Create Function-based Age-based (Leslie) Matrix Projection Model*

Description

Function `fleslie()` returns age-based (Leslie) MPMs corresponding to the patches and occasions given, including the associated component transition and fecundity matrices, data frames detailing the characteristics of the exact ages corresponding to rows and columns in estimated matrices, and a data frame characterizing the patch and occasion combinations corresponding to these matrices.

Usage

```
fleslie(
  year = "all",
  patch = NULL,
  prebreeding = TRUE,
  data = NULL,
```

```
  modelsuite = NULL,
  surv_model = NULL,
  fec_model = NULL,
  paramnames = NULL,
  supplement = NULL,
  start_age = NA,
  last_age = NA,
  fecage_min = NA,
  fecage_max = NA,
  continue = TRUE,
  inda = NULL,
  indb = NULL,
  indc = NULL,
  annua = NULL,
  annub = NULL,
  annuc = NULL,
  surv_dev = 0,
  fec_dev = 0,
  density = NA,
  fecmod = 1,
  random.inda = FALSE,
  random.indb = FALSE,
  random.indc = FALSE,
  negfec = FALSE,
  reduce = FALSE,
  simple = FALSE,
  err_check = FALSE,
  exp_tol = 700,
  theta_tol = 1e+08,
  sparse_output = FALSE
)
```

Arguments

year	A variable corresponding to observation occasion, or a set of such values, given in values associated with the year term used in linear model development. Defaults to "all", in which case matrices will be estimated for all occasions.
patch	A variable designating which patches or subpopulations will have matrices estimated. Defaults to "all", but can also be set to specific patch names or a vector thereof.
prebreeding	A logical value indicating whether the life history model is a pre-breeding model. Defaults to TRUE.
data	The historical vertical demographic data frame used to estimate vital rates (class hfvdata). The original data frame is generally required in order to initialize occasions and patches properly, and to assess the range of ages observed in the population. Not required if option modelsuite is set to a vrm_input object.
modelsuite	One of three kinds of lists. The first is a lefkoMod object holding the vital rate models and associated metadata. The second is a lefkoModList ob-

ject, which is a list of lefkoMod objects generally created to conduct a bootstrapped MPM analysis. Alternatively, an object of class vrm_input may be provided. If given, then surv_model, obs_model, size_model, sizeb_model, sizec_model, repst_model, fec_model, jsurv_model, jobs_model, jsize_model, jsizeb_model, jsizec_model, jrepst_model, jmatst_model, and paramnames are not required. One or more of these models should include size or reproductive status in occasion *t*-1. Although this is optional input, it is recommended, and without it all vital rate model inputs (named XX_model) are required.

surv_model — A linear model predicting survival probability. This can be a model of class glm or glmer, and requires a predicted binomial variable under a logit link. Ignored if modelsuite is provided. This model must have been developed in a modeling exercise testing only the impacts of occasion *t*.

fec_model — A linear model predicting fecundity. This can be a model of class glm, glmer, glmmTMB, zeroinfl, vglm, lm, or lmer. Ignored if modelsuite is provided. This model must have been developed in a modeling exercise testing only the impacts of occasion *t*.

paramnames — A data frame with three columns, the first describing all terms used in linear modeling, the second (must be called mainparams) giving the general model terms that will be used in matrix creation, and the third showing the equivalent terms used in modeling (must be named modelparams). Function create_pm() can be used to create a skeleton paramnames object, which can then be edited. Only required if modelsuite is not supplied.

supplement — An optional data frame of class lefkoSD that provides supplemental data that should be incorporated into the MPM. Three kinds of data may be integrated this way: transitions to be estimated via the use of proxy transitions, transition overwrites from the literature or supplemental studies, and transition multipliers for survival and fecundity. This data frame should be produced using the supplemental() function.

start_age — The age from which to start the matrix. Defaults to NA, in which case age 1 is used if prebreeding = TRUE, and age 0 is used if prebreeding = FALSE.

last_age — The final age to use in the matrix. Defaults to NA, in which case the highest age in the dataset is used.

fecage_min — The minimum age at which reproduction is possible. Defaults to NA, which is interpreted to mean that fecundity should be assessed starting in the minimum age observed in the dataset.

fecage_max — The maximum age at which reproduction is possible. Defaults to NA, which is interpreted to mean that fecundity should be assessed until the final observed age.

continue — A logical value designating whether to allow continued survival of individuals past the final age noted in the stageframe, using the demographic characteristics of the final age. Defaults to TRUE.

inda — Can be a single value to use for individual covariate a in all matrices, or a vector of such values corresponding to each occasion in the dataset. Defaults to NULL.

indb — Can be a single value to use for individual covariate b in all matrices, or a vector of such values corresponding to each occasion in the dataset. Defaults to NULL.

indc	Can be a single value to use for individual covariate c in all matrices, or a vector of such values corresponding to each occasion in the dataset. Defaults to NULL.
annua	Can be a single value to use for annual covariate a in all matrices, a pair of values to use for times *t* and *t*-1 in historical matrices, or a vector of such values corresponding to each occasion in the dataset. Defaults to NULL.
annub	Can be a single value to use for annual covariate b in all matrices, a pair of values to use for times *t* and *t*-1 in historical matrices, or a vector of such values corresponding to each occasion in the dataset. Defaults to NULL.
annuc	Can be a single value to use for annual covariate c in all matrices, a pair of values to use for times *t* and *t*-1 in historical matrices, or a vector of such values corresponding to each occasion in the dataset. Defaults to NULL.
surv_dev	A numeric value to be added to the y-intercept in the linear model for survival probability. Defaults to 0.
fec_dev	A numeric value to be added to the y-intercept in the linear model for fecundity. Defaults to 0.
density	A numeric value indicating density value to use to propagate matrices. Only needed if density is an explanatory term used in linear models. Defaults to NA.
fecmod	A scalar multiplier of fecundity. Defaults to 1.0.
random.inda	A logical value denoting whether to treat individual covariate a as a random, categorical variable. Otherwise is treated as a fixed, numeric variable. Defaults to FALSE.
random.indb	A logical value denoting whether to treat individual covariate b as a random, categorical variable. Otherwise is treated as a fixed, numeric variable. Defaults to FALSE.
random.indc	A logical value denoting whether to treat individual covariate c as a random, categorical variable. Otherwise is treated as a fixed, numeric variable. Defaults to FALSE.
negfec	A logical value denoting whether fecundity values estimated to be negative should be reset to 0. Defaults to FALSE.
reduce	A logical value denoting whether to remove ages associated solely with 0 transitions. These are only removed in cases where the associated row and column sums in ALL matrices estimated equal 0. Defaults to FALSE, and should generally not be used in age-based MPMs.
simple	A logical value indicating whether to produce A, U, and F matrices, or only the latter two. Defaults to FALSE, in which case all three are output.
err_check	A logical value indicating whether to append extra information used in matrix calculation within the output list. 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.
sparse_output	A logical value indicating whether to output matrices in sparse format. Defaults to FALSE, in which case all matrices are output in standard matrix format.

Value

If the user inputs a standard lefkoMod or vrm_input object in argument modelsuite, or individual vital rate models are input separately,then this function will return an object of class lefkoMat. If the user inputs an object of class lefkoModList in argument modelsuite, then the output will be an object of class lefkoMatList, in which each element is an object of class lefkoMat.

A lefkoMat object is a list that holds one full matrix projection model and all of its metadata. The structure has the following elements:

A	A list of full projection matrices in order of sorted patches and occasions. All matrices output in R's matrix class, or in the dgCMatrix class from the Matrix package if sparse.
U	A list of survival transition matrices sorted as in A. All matrices output in R's matrix class, or in the dgCMatrix class from the Matrix package if sparse.
F	A list of fecundity matrices sorted as in A. All matrices output in R's matrix class, or in the dgCMatrix class from the Matrix package if sparse.
hstages	Set to NA for Leslie MPMs.
agestages	Set to NA for Leslie MPMs.
ahstages	A data frame detailing the characteristics of associated ages, in the form of a modified stageframe including reproduction status.
labels	A data frame giving the patch and year of each matrix in order. In fleslie(), only one population may be analyzed at once.
dataqc	A vector showing the numbers of individuals and rows in the vertical dataset used as input.
matrixqc	A short vector describing the number of non-zero elements in U and F matrices, and the number of annual matrices.
modelqc	This is the qc portion of the modelsuite input.
prob_out	An optional element only added if err_check = TRUE. This is a list of vital rate probability matrices, with 7 columns in the order of survival, observation probability, reproduction probability, primary size transition probability, secondary size transition probability, tertiary size transition probability, and probability of juvenile transition to maturity.

Notes

Unlike rlefko2(), rlefko3(), arlefko2(), and rleslie(), this function does not currently distinguish populations.

This function will yield incorrect estimates if the models utilized incorporate state in occasion *t*-1, or any size or reproductive status terms.

Users may at times wish to estimate MPMs using a dataset incorporating multiple patches or subpopulations, but without discriminating between those patches or subpopulations. Should the aim of analysis be a general MPM that does not distinguish these patches or subpopulations, the modelsearch() run should not include patch terms.

Input options including multiple variable names must be entered in the order of variables in occasion *t*+1 and *t*. Rearranging the order will lead to erroneous calculations, and may lead to fatal errors.

Care should be taken to match the random status of year and patch to the states of those variables within the modelsuite. If they do not match, then they will be treated as zeroes in vital rate estimation.

Individual covariates are treated as categorical only if they are set as random terms. Fixed categorical individual covariates are currently not allowed. However, such terms may be supplied if the `modelsuite` option is set to a `vrm_input` object. In that case, the user should also set the logical random switch for the individual covariate to be used to `TRUE` (e.g., `random.inda = TRUE`).

See Also

`mpm_create()`

`flefko3()`

`flefko2()`

`aflefko2()`

`arlefko2()`

`rlefko3()`

`rlefko2()`

`rleslie()`

Examples

```
data(lathyrus)

lathvert_base <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  sizeacol = "Volume88", repstracol = "FCODE88", fecacol = "Intactseed88",
  deadacol = "Dead1988", censorcol = "Missing1988", censorkeep = NA,
  censor = TRUE, NAas0 = TRUE, NRasRep = TRUE, NOasObs = TRUE)

lathvert_base$feca3 <- round(lathvert_base$feca3)
lathvert_base$feca2 <- round(lathvert_base$feca2)
lathvert_base$feca1 <- round(lathvert_base$feca1)

lathvert_age <- subset(lathvert_base, firstseen > 1988)

lath_survival <- glm(alive3 ~ obsage + as.factor(year2), data = lathvert_age,
  family = "binomial")
lath_fecundity <- glm(feca2 ~ obsage + as.factor(year2), data = lathvert_age,
  family = "poisson")

mod_params <- create_pm(name_terms = TRUE)
mod_params$modelparams[22] <- "obsage"

lathmat2fleslie <- fleslie(year = "all", data = lathvert_age,
  surv_model = lath_survival, fec_model = lath_fecundity,
  paramnames = mod_params, fecage_min = 1)
```

f_projection3 *Project Function-based Matrix Projection Model*

Description

Function f_projection3() develops and projects function-based matrix models. Unlike projection3(), which uses matrices provided as input via already created lefkoMat objects, function f_projection3() creates matrices at each time step from vital rate models and parameter inputs provided. Projections may be stochastic or not, and may be density dependent in either case. Also handles replication.

Usage

```
f_projection3(
  format,
  prebreeding = TRUE,
  start_age = NA_integer_,
  last_age = NA_integer_,
  fecage_min = NA_integer_,
  fecage_max = NA_integer_,
  cont = TRUE,
  stochastic = FALSE,
  standardize = FALSE,
  growthonly = TRUE,
  repvalue = FALSE,
  integeronly = FALSE,
  substoch = 0L,
  ipm_cdf = TRUE,
  nreps = 1L,
  times = 10000L,
  repmod = 1,
  exp_tol = 700,
  theta_tol = 1e+08,
  random_inda = FALSE,
  random_indb = FALSE,
  random_indc = FALSE,
  err_check = FALSE,
  quiet = FALSE,
  data = NULL,
  stageframe = NULL,
  supplement = NULL,
  repmatrix = NULL,
  overwrite = NULL,
  modelsuite = NULL,
  paramnames = NULL,
  year = NULL,
  patch = NULL,
  sp_density = NULL,
```

```
  ind_terms = NULL,
  ann_terms = NULL,
  dev_terms = NULL,
  surv_model = NULL,
  obs_model = NULL,
  size_model = NULL,
  sizeb_model = NULL,
  sizec_model = NULL,
  repst_model = NULL,
  fec_model = NULL,
  jsurv_model = NULL,
  jobs_model = NULL,
  jsize_model = NULL,
  jsizeb_model = NULL,
  jsizec_model = NULL,
  jrepst_model = NULL,
  jmatst_model = NULL,
  start_vec = NULL,
  start_frame = NULL,
  tweights = NULL,
  density = NULL,
  density_vr = NULL,
  stage_weights = NULL,
  sparse = NULL
)
```

Arguments

format	An integer indicating the kind of function-based MPM to create. Possible choices include: 1, Ehrlen-format historical MPM; 2, deVries-format historical MPM; 3, ahistorical MPM; 4, age-by-stage MPM; and 5, Leslie (age-based) MPM.
prebreeding	A logical value indicating whether the life history model is a pre-breeding model. Only used in Leslie and age-by-stage MPMs. Defaults to TRUE.
start_age	The age from which to start the matrix. Defaults to NA, in which case age 1 is used if prebreeding = TRUE, and age 0 is used if prebreeding = FALSE.
last_age	The final age to use in the matrix. Defaults to NA, in which case the highest age in the dataset is used.
fecage_min	The minimum age at which reproduction is possible. Defaults to NA, which is interpreted to mean that fecundity should be assessed starting in the minimum age observed in the dataset.
fecage_max	The maximum age at which reproduction is possible. Defaults to NA, which is interpreted to mean that fecundity should be assessed until the final observed age.
cont	A logical value designating whether to allow continued survival of individuals past the final age noted in the stageframe, using the demographic characteristics of the final age. Defaults to TRUE.

stochastic	A logical value denoting whether to conduct a stochastic projection or a deterministic / cyclical projection.
standardize	A logical value denoting whether to re-standardize the population size to 1.0 at each occasion. Used in density-independent simulations in which it is more important to know the general trend in population growth than the explicit growth rate. Defaults to FALSE.
growthonly	A logical value indicating whether to produce only the projected population size at each occasion (TRUE), or also to produce vectors showing the stage distribution at each occasion (FALSE). Defaults to TRUE.
repvalue	A logical value indicating whether to calculate reproductive value vectors at each time step. Can only be set to TRUE if growthonly = FALSE. Setting to TRUE may dramatically increase the duration of calculations. Defaults to FALSE.
integeronly	A logical value indicating whether to round the number of individuals projected in each stage at each occasion 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.
ipm_cdf	A logical value indicating whether to estimate size transitions using the cumulative density function in cases with continuous distributions. Defaults to TRUE, with the midpoint method used if FALSE.
nreps	The number of replicate projections. Defaults to 1.
times	Number of occasions to iterate per replicate. Defaults to 10000.
repmod	A scalar multiplier of fecundity. Defaults to 1.
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.
random_inda	A logical value denoting whether to treat individual covariate a as a random, categorical variable. Otherwise is treated as a fixed, numeric variable. Defaults to FALSE.
random_indb	A logical value denoting whether to treat individual covariate b as a random, categorical variable. Otherwise is treated as a fixed, numeric variable. Defaults to FALSE.
random_indc	A logical value denoting whether to treat individual covariate c as a random, categorical variable. Otherwise is treated as a fixed, numeric variable. Defaults to FALSE.
err_check	A logical value indicating whether to append extra output for debugging purposes. Defaults to FALSE.

quiet — A logical value indicating whether warning messages should be suppressed. Defaults to FALSE.

data — The historical vertical demographic data frame used to estimate vital rates (class hfvdata), which is required to initialize times and patches properly. Variable names should correspond to the naming conventions in verticalize3() and historicalize3(). If attempting bootstrapped projection, then an object of class hfvlist is required.

stageframe — An object of class stageframe. These objects are generated by function sf_create(), and include information on the size, observation status, propagule status, reproduction status, immaturity status, maturity status, stage group, size bin widths, and other key characteristics of each ahistorical stage. Required for all MPM formats except Leslie MPMs.

supplement — An optional data frame of class lefkoSD that provides supplemental data that should be incorporated into the MPM. Three kinds of data may be integrated this way: transitions to be estimated via the use of proxy transitions, transition overwrites from the literature or supplemental studies, and transition multipliers for survival and fecundity. This data frame should be produced using the supplemental() function. Can be used in place of or in addition to an overwrite table (see overwrite below) and a reproduction matrix (see repmatrix below).

repmatrix — An optional reproduction matrix. This matrix is composed mostly of 0s, with non-zero entries acting as element identifiers and multipliers for fecundity (with 1 equaling full fecundity). If left blank, and no supplement is provided, then flefko3() will assume that all stages marked as reproductive produce offspring at 1x that of estimated fecundity, and that offspring production will yield the first stage noted as propagule or immature. May be the dimensions of either a historical or an ahistorical matrix. If the latter, then all stages will be used in occasion *t*-1 for each suggested ahistorical transition.

overwrite — An optional data frame developed with the overwrite() function describing transitions to be overwritten either with given values or with other estimated transitions. Note that this function supplements overwrite data provided in supplement.

modelsuite — A lefkoMod object, at minimum with all required best-fit vital rate models and a paramnames data frame, and following the naming conventions used in this package. If given, then surv_model, obs_model, size_model, sizeb_model, sizec_model, repst_model, fec_model, jsurv_model, jobs_model, jsize_model, jsizeb_model, jsizec_model, jrepst_model, jmatst_model, paramnames, yearcol, and patchcol are not required. Alternatively, an object of class vrm_input, serving the same role. Although this is optional input, it is recommended, and without it separate vital rate model inputs (named XX_model) are required. If conducting bootstrapped projection, then an object of lefkoModList is required.

paramnames — A data frame with three columns, the first describing all terms used in linear modeling, the second (must be called mainparams) giving the general model terms that will be used in matrix creation, and the third showing the equivalent terms used in modeling (must be named modelparams). Function create_pm()

can be used to create a skeleton paramnames object, which can then be edited. Only required if modelsuite is not supplied.

year — Either a single integer value corresponding to the year to project, or a vector of times elements with the year to use at each time step. Defaults to NA, in which the first year in the set of years in the dataset is projected. If a vector shorter than times is supplied, then this vector will be cycled.

patch — A value of NA, a single string value corresponding to the patch to project, or a vector of times elements with the patch to use at each time step. If a vector shorter than times is supplied, then this vector will be cycled. Note that this function currently does not handle multiple projections for different patches in the same run.

sp_density — Either a single numeric value of spatial density to use in vital rate models in all time steps, or a vector of times elements of such numeric values. Defaults to NA.

ind_terms — An optional data frame with 3 columns and times rows giving the values of individual covariates a, b, and c, respectively, for each projected time. Unused terms must be set to 0 (use of NA will produce errors.)

ann_terms — An optional data frame with 3 columns and times rows giving the values of annual covariates a, b, and c, respectively, for each projected time. Unused terms must be set to 0 (use of NA will produce errors.)

dev_terms — An optional data frame with 14 columns and 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.)

surv_model — A linear model predicting survival probability. This can be a model of class glm or glmer, and requires a predicted binomial variable under a logit link. Ignored if modelsuite is provided. This model must have been developed in a modeling exercise testing the impacts of occasions t and t-1.

obs_model — A linear model predicting sprouting or observation probability. This can be a model of class glm or glmer, and requires a predicted binomial variable under a logit link. Ignored if modelsuite is provided. This model must have been developed in a modeling exercise testing the impacts of occasions t and t-1.

size_model — A linear model predicting primary size. This can be a model of class glm, glmer, glmmTMB, zeroinfl, vglm, lm, or lmer. Ignored if modelsuite is provided. This model must have been developed in a modeling exercise testing the impacts of occasions t and t-1.

sizeb_model — A linear model predicting secondary size. This can be a model of class glm, glmer, glmmTMB, zeroinfl, vglm, lm, or lmer. Ignored if modelsuite is provided. This model must have been developed in a modeling exercise testing the impacts of occasions t and t-1.

sizec_model — A linear model predicting tertiary size. This can be a model of class glm, glmer, glmmTMB, zeroinfl, vglm, lm, or lmer. Ignored if modelsuite is provided.

This model must have been developed in a modeling exercise testing the impacts of occasions *t* and *t*-1.

`repst_model` A linear model predicting reproduction probability. This can be a model of class `glm` or `glmer`, and requires a predicted binomial variable under a logit link. Ignored if `modelsuite` is provided. This model must have been developed in a modeling exercise testing the impacts of occasions *t* and *t*-1.

`fec_model` A linear model predicting fecundity. This can be a model of class `glm`, `glmer`, `glmmTMB`, `zeroinfl`, `vglm`, `lm`, or `lmer`. Ignored if `modelsuite` is provided. This model must have been developed in a modeling exercise testing the impacts of occasions *t* and *t*-1.

`jsurv_model` A linear model predicting juvenile survival probability. This can be a model of class `glm` or `glmer`, and requires a predicted binomial variable under a logit link. Ignored if `modelsuite` is provided. This model must have been developed in a modeling exercise testing the impacts of occasions *t* and *t*-1.

`jobs_model` A linear model predicting juvenile sprouting or observation probability. This can be a model of class `glm` or `glmer`, and requires a predicted binomial variable under a logit link. Ignored if `modelsuite` is provided. This model must have been developed in a modeling exercise testing the impacts of occasions *t* and *t*-1.

`jsize_model` A linear model predicting juvenile primary size. This can be a model of class `glm`, `glmer`, `glmmTMB`, `zeroinfl`, `vglm`, `lm`, or `lmer`. Ignored if `modelsuite` is provided. This model must have been developed in a modeling exercise testing the impacts of occasions *t* and *t*-1.

`jsizeb_model` A linear model predicting juvenile secondary size. This can be a model of class `glm`, `glmer`, `glmmTMB`, `zeroinfl`, `vglm`, `lm`, or `lmer`. Ignored if `modelsuite` is provided. This model must have been developed in a modeling exercise testing the impacts of occasions *t* and *t*-1.

`jsizec_model` A linear model predicting juvenile tertiary size. This can be a model of class `glm`, `glmer`, `glmmTMB`, `zeroinfl`, `vglm`, `lm`, or `lmer`. Ignored if `modelsuite` is provided. This model must have been developed in a modeling exercise testing the impacts of occasions *t* and *t*-1.

`jrepst_model` A linear model predicting reproduction probability of a mature individual that was immature in time *t*. This can be a model of class `glm` or `glmer`, and requires a predicted binomial variable under a logit link. Ignored if `modelsuite` is provided. This model must have been developed in a modeling exercise testing the impacts of occasions *t* and *t*-1.

`jmatst_model` A linear model predicting maturity probability of an individual that was immature in time *t*. This can be a model of class `glm` or `glmer`, and requires a predicted binomial variable under a logit link. Ignored if `modelsuite` is provided. This model must have been developed in a modeling exercise testing the impacts of occasions *t* and *t*-1.

`start_vec` An optional numeric vector denoting the starting stage distribution for the projection. Defaults to a single individual of each stage.

`start_frame` An optional data frame characterizing stages, age-stages, or stage-pairs that should be set to non-zero values in the starting vector, and what those values should be. Can only be used with `lefkoMat` objects.

`tweights` An optional numeric vector or matrix denoting the probabilities of choosing each matrix in a stochastic projection. If a matrix is input, 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 a vector is input, then the choice of annual matrix is assumed to be independent of the current matrix. Defaults to equal weighting among matrices.

`density` An optional data frame describing the matrix elements that will be subject to density dependence, and the exact kind of density dependence that they will be subject to, or a list of such objects. The data frame used should be an object of class `lefkoDens`, which is the output from function `density_input()`.

`density_vr` An optional data frame describing density dependence relationships in vital rates, if such relationships are to be assumed. The data frame must be of class `lefkoDensVR`, which is the output from the function `density_vr()`.

`stage_weights` An optional object of class `lefkoEq` giving the degree to which individuals in each stage are equivalent to one another, or a list of such objects. May also be a numeric vector (or a list thereof), 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.

`sparse` A text string indicating whether to use sparse matrix encoding (`"yes"`) or dense matrix encoding (`"no"`). Defaults to `"auto"`, in which case sparse matrix encoding is used with square matrices with at least 50 rows and no more than 50% of elements with values greater than zero. Can also be entered as a logical value if forced sparse (`TRUE`) or forced dense (`FALSE`) projection is desired.

Value

If running a single (including replicated) projection, then the output is a list of class `lefkoProj`, which always includes the first three elements of the following, and also includes the remaining elements below when a `lefkoMat` object is used as input:

`projection` A list of lists of matrices showing the total number of individuals per stage per occasion. The first list corresponds to each pop-patch followed by each population (this top-level list is a single element in `f_projection3()`). The inner list corresponds to replicates within each pop-patch or population.

`stage_dist` A list of lists of the actual stage distribution in each occasion in each replicate in each pop-patch or population. The list structure is the same as in `projection3()`.

`rep_value` A list of lists of the actual reproductive value in each occasion in each replicate in each pop-patch or population. The list structure is the same as in `projection3()`.

`pop_size` A list of matrices showing the total population size in each occasion per replicate (row within matrix) per pop-patch or population (list element). Only a single pop-patch or population is allowed in `f_projection3()`.

`labels` A data frame showing the order of populations and patches in item `projection`.

`ahstages` The original stageframe used in the study.

`hstages` A data frame showing the order of historical stage pairs.

agestages	A data frame showing the order of age-stage pairs.
labels	A short data frame indicating the population (always 1), and patch (either the numeric index of the single chosen patch, or 1 in all other cases).
control	A short vector indicating the number of replicates and the number of occasions projected per replicate.
density	The data frame input under the density option. Only provided if input by the user.
density_vr	The data frame input under the density_vr option. Only provided if input by the user.

If running bootstrapped projections (i.e. using hfvlist and lefkoModList input), then will output an object of class lefkoProjList, which is just a simple list of lefkoProj objects.

Notes

Population projection can be a very time-consuming activity, and it is most time-consuming when matrices need to be created at each time step. We have created this function to work as quickly as possible, but some options will slow analysis. First, the err_check option should always be set to FALSE, as the added created output will not only slow the analysis down but also potentially crash the memory if matrices are large enough. Second, the repvalue option should be set to FALSE unless reproductive values are genuinely needed, since this step requires concurrent backward projection and so in some cases may double total run time. Next, if the only needed data is the total population size and age/stage structure at each time step, then setting growthonly = TRUE will yield the quickest possible run time. Finally, the default behavior of the function is to round down fractional values of individuals, and to stop running projections (replicates) when the population drops to 0. Setting integeronly = FALSE will have the impact of increasing runtime, potentially dramatically, since the population can reach a point in which the population size is extremely small but not equal to 0.

Projections with large matrices may take a long time to run. To assess the likely running time, try using a low number of iterations on a single replicate first. For example, set nreps = 1 and times = 10 for a trial run. If a full run is set and takes too long, press the STOP button in RStudio to cancel the projection run, or click esc.

This function currently allows three forms of density dependence. The first modifies matrix elements on the basis of the input provided in option density, and so alters matrix elements once the matrix has already been created. The second form alters the vital rates estimated, and so estimates matrix elements using vital rate values already modified by density. This second form uses the input provided in option density_vr. These two forms of density dependence utilize the projected population size at some time to make these alterations. The third form of density dependence also alters the vital rates, but using spatial density supplied via option sp_density and only in vital rates in which spatial density is included as a fixed factor in the associated vital rate model.

When running density dependent simulations involving user-set exponents, such as the beta term in the Ricker function and both the alpha and beta terms in the Usher function, values above or below the computer limits may cause unpredictable behavior. Noted odd behavior includes sudden shifts in population size to negative values. This function produces warnings when such values are used, and the values used for warnings may be reset with the exp_tol term. In addition, this function resets beta values for the Ricker function automatically to positive or negative exp_tol, giving a warning when doing so.

Consistently positive population growth can quickly lead to population size numbers larger than can be handled computationally. In that circumstance, a continuously rising population size will suddenly become NaN for the remainder of the projection.

This function does not reduce the dimensionality of matrices developed for projection.

Speed can sometimes be increased by shifting from automatic sparse matrix determination to forced dense or sparse matrix projection. This will most likely occur when matrices have between 30 and 300 rows and columns. Defaults work best when matrices are very small and dense, or very large and sparse.

Some issues may arise in first-order Markovian stochastic projections if the year argument is used. Use the matrix input in the tweights argument to eliminate any years from consideration that are not needed.

See Also

start_input()

density_input()

density_vr()

projection3()

flefko3()

flefko2()

aflefko2()

fleslie()

append_lP()

summary.lefkoProj()

plot.lefkoProj()

Examples

```
data(lathyrus)

sizevector <- c(0, 4.6, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 1, 2, 3, 4, 5, 6, 7, 8,
  9)
stagevector <- c("Sd", "Sdl", "Dorm", "Sz1nr", "Sz2nr", "Sz3nr", "Sz4nr",
  "Sz5nr", "Sz6nr", "Sz7nr", "Sz8nr", "Sz9nr", "Sz1r", "Sz2r", "Sz3r",
  "Sz4r", "Sz5r", "Sz6r", "Sz7r", "Sz8r", "Sz9r")
repvector <- c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1)
obsvector <- c(0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
matvector <- c(0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0)
indataset <- c(0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 4.6, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
  0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5)

lathframeln <- sf_create(sizes = sizevector, stagenames = stagevector,
```

```
  repstatus = repvector, obsstatus = obsvector, matstatus = matvector,
  immstatus = immvector, indataset = indataset, binhalfwidth = binvec,
  propstatus = propvector)

lathvertln <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "lnVol88", repstracol = "Intactseed88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframeln, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, NAas0 = TRUE, censor = TRUE)

lathvertln$feca2 <- round(lathvertln$feca2)
lathvertln$feca1 <- round(lathvertln$feca1)
lathvertln$feca3 <- round(lathvertln$feca3)

lathvertln_adults <- subset(lathvertln, stage2index > 2)
surv_model <- glm(alive3 ~ sizea2 + sizea1 + as.factor(patchid) +
  as.factor(year2), data = lathvertln_adults, family = "binomial")

obs_data <- subset(lathvertln_adults, alive3 == 1)
obs_model <- glm(obsstatus3 ~ as.factor(patchid), data = obs_data,
  family = "binomial")

size_data <- subset(obs_data, obsstatus3 == 1)
siz_model <- lm(sizea3 ~ sizea2 + sizea1 + repstatus1 + as.factor(patchid) +
  as.factor(year2), data = size_data)

reps_model <- glm(repstatus3 ~ sizea2 + sizea1 + as.factor(patchid) +
  as.factor(year2), data = size_data, family = "binomial")

fec_data <- subset(lathvertln_adults, repstatus2 == 1)
fec_model <- glm(feca2 ~ sizea2 + sizea1 + repstatus1 + as.factor(patchid),
  data = fec_data, family = "poisson")

lathvertln_juvs <- subset(lathvertln, stage2index < 3)
jsurv_model <- glm(alive3 ~ as.factor(patchid), data = lathvertln_juvs,
  family = "binomial")

jobs_data <- subset(lathvertln_juvs, alive3 == 1)
jobs_model <- glm(obsstatus3 ~ 1, family = "binomial", data = jobs_data)

jsize_data <- subset(jobs_data, obsstatus3 == 1)
jsiz_model <- lm(sizea3 ~ as.factor(year2), data = jsize_data)

jrepst_model <- 0
jmatst_model <- 1

mod_params <- create_pm(name_terms = TRUE)
mod_params$modelparams[3] <- "patchid"
mod_params$modelparams[4] <- "alive3"
mod_params$modelparams[5] <- "obsstatus3"
mod_params$modelparams[6] <- "sizea3"
mod_params$modelparams[9] <- "repstatus3"
```

```
mod_params$modelparams[11] <- "feca2"
mod_params$modelparams[12] <- "sizea2"
mod_params$modelparams[13] <- "sizea1"
mod_params$modelparams[18] <- "repstatus2"
mod_params$modelparams[19] <- "repstatus1"

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "mat", "Sd", "Sdl"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "Sdl", "rep", "rep"),
  stage1 = c("Sd", "rep", "Sd", "rep", "Sd", "mat", "mat"),
  eststage3 = c(NA, NA, NA, NA, "mat", NA, NA),
  eststage2 = c(NA, NA, NA, NA, "Sdl", NA, NA),
  eststage1 = c(NA, NA, NA, NA, "Sdl", NA, NA),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, NA, 0.345, 0.054),
  type = c(1, 1, 1, 1, 1, 3, 3), type_t12 = c(1, 2, 1, 2, 1, 1, 1),
  stageframe = lathframeln, historical = TRUE)

# While we do not use MPMs to initialize f_projections3(), we do use MPMs to
# initialize functions start_input() and density_input().
lathmat3ln <- flefko3(year = "all", patch = "all", data = lathvertln,
  stageframe = lathframeln, supplement = lathsupp3, paramnames = mod_params,
  surv_model = surv_model, obs_model = obs_model, size_model = siz_model,
  repst_model = reps_model, fec_model = fec_model, jsurv_model = jsurv_model,
  jobs_model = jobs_model, jsize_model = jsiz_model,
  jrepst_model = jrepst_model, jmatst_model = jmatst_model, reduce = FALSE)

e3m_sv <- start_input(lathmat3ln, stage2 = "Sd", stage1 = "Sd", value = 1000)

dyn7 <- c(TRUE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE,
  FALSE, FALSE, FALSE, FALSE, FALSE)
dst7 <- c(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
dal7 <- c(0.5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
dbe7 <- c(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)

e3d_vr <- density_vr(density_yn = dyn7, style = dst7, alpha = dal7,
  beta = dbe7)

trial7_dvr_1 <- f_projection3(format = 1, data = lathvertln, supplement = lathsupp3,
  paramnames = mod_params, stageframe = lathframeln, nreps = 2,
  surv_model = surv_model, obs_model = obs_model, size_model = siz_model,
  repst_model = reps_model, fec_model = fec_model, jsurv_model = jsurv_model,
  jobs_model = jobs_model, jsize_model = jsiz_model,
  jrepst_model = jrepst_model, jmatst_model = jmatst_model,
  times = 100, stochastic = TRUE, standardize = FALSE, growthonly = TRUE,
  substoch = 0, sp_density = 0, start_frame = e3m_sv, density_vr = e3d_vr)
```

---

hfv_qc *Assess Quality of hfv Datasets*

Description

Function `hfv_qc()` tests the overall quality of hfv datasets, and also runs a series of tests to assess which statistical distributions match the variables within these datasets. The input format is equivalent to the input format of function `modelsearch()`, allowing users to assess vital rate variable distributions assuming the same internal dataset subsetting used by the latter function and simply copy and pasting the parameter options from one function to the other.

Usage

```
hfv_qc(
  data,
  stageframe = NULL,
  historical = TRUE,
  suite = "size",
  vitalrates = c("surv", "size", "fec"),
  surv = c("alive3", "alive2", "alive1"),
  obs = c("obsstatus3", "obsstatus2", "obsstatus1"),
  size = c("sizea3", "sizea2", "sizea1"),
  sizeb = c(NA, NA, NA),
  sizec = c(NA, NA, NA),
  repst = c("repstatus3", "repstatus2", "repstatus1"),
  fec = c("feca3", "feca2", "feca1"),
  stage = c("stage3", "stage2", "stage1"),
  matstat = c("matstatus3", "matstatus2", "matstatus1"),
  indiv = "individ",
  patch = NA,
  year = "year2",
  density = NA,
  patch.as.random = TRUE,
  year.as.random = TRUE,
  juvestimate = NA,
  juvsize = FALSE,
  fectime = 2,
  censor = NA,
  age = NA,
  indcova = NA,
  indcovb = NA,
  indcovc = NA,
  random.indcova = FALSE,
  random.indcovb = FALSE,
  random.indcovc = FALSE,
  test.group = FALSE,
  ...
)
```

Arguments

data	The vertical dataset to be used for analysis. This dataset should be of class hfvdata, but can also be a data frame formatted similarly to the output format provided by functions verticalize3() or historicalize3(), as long as all needed variables are properly designated.
stageframe	The stageframe characterizing the life history model used. Optional unless test.group = TRUE, in which case it is required. Defaults to NULL.
historical	A logical variable denoting whether to assess the effects of state in occasion *t*-1, in addition to state in occasion *t*. Defaults to TRUE.
suite	This describes the global model for each vital rate estimation, and has the following possible values: full, includes main effects and all two-way interactions of size and reproductive status; main, includes main effects only of size and reproductive status; size, includes only size (also interactions between size in historical model); rep, includes only reproductive status (also interactions between status in historical model); age, all vital rates estimated with age and y-intercepts only; cons, all vital rates estimated only as y-intercepts. Defaults to size.
vitalrates	A vector describing which vital rates will be estimated via linear modeling, with the following options: surv, survival probability; obs, observation probability; size, overall size; repst, probability of reproducing; and fec, amount of reproduction (overall fecundity). May also be set to vitalrates = "leslie", which is equivalent to setting c("surv", "fec") for a Leslie MPM. This choice also determines how internal data subsetting for vital rate model estimation will work. Defaults to c("surv", "size", "fec").
surv	A vector indicating the variable names coding for status as alive or dead in occasions *t*+1, *t*, and *t*-1, respectively. Defaults to c("alive3", "alive2", "alive1").
obs	A vector indicating the variable names coding for observation status in occasions *t*+1, *t*, and *t*-1, respectively. Defaults to c("obsstatus3", "obsstatus2", "obsstatus1").
size	A vector indicating the variable names coding for the primary size variable on occasions *t*+1, *t*, and *t*-1, respectively. Defaults to c("sizea3", "sizea2", "sizea1").
sizeb	A vector indicating the variable names coding for the secondary size variable on occasions *t*+1, *t*, and *t*-1, respectively. Defaults to c(NA, NA, NA), in which case sizeb is not used.
sizec	A vector indicating the variable names coding for the tertiary size variable on occasions *t*+1, *t*, and *t*-1, respectively. Defaults to c(NA, NA, NA), in which case sizec is not used.
repst	A vector indicating the variable names coding for reproductive status in occasions *t*+1, *t*, and *t*-1, respectively. Defaults to c("repstatus3", "repstatus2", "repstatus1").
fec	A vector indicating the variable names coding for fecundity in occasions *t*+1, *t*, and *t*-1, respectively. Defaults to c("feca3", "feca2", "feca1").

`stage` A vector indicating the variable names coding for stage in occasions *t*+1, *t*, and *t*-1. Defaults to `c("stage3", "stage2", "stage1")`.

`matstat` A vector indicating the variable names coding for maturity status in occasions *t*+1, *t*, and *t*-1. Defaults to `c("matstatus3", "matstatus2", "matstatus1")`.

`indiv` A text value indicating the variable name coding individual identity. Defaults to `"individ"`.

`patch` A text value indicating the variable name coding for patch, where patches are defined as permanent subgroups within the study population. Defaults to `NA`.

`year` A text value indicating the variable coding for observation occasion *t*. Defaults to `"year2"`.

`density` A text value indicating the name of the variable coding for spatial density, should the user wish to test spatial density as a fixed factor affecting vital rates. Defaults to `NA`.

`patch.as.random`
If set to `TRUE` and `approach = "mixed"`, then `patch` is included as a random factor. If set to `FALSE` and `approach = "glm"`, then `patch` is included as a fixed factor. All other combinations of logical value and `approach` lead to `patch` not being included in modeling. Defaults to `TRUE`.

`year.as.random` If set to `TRUE` and `approach = "mixed"`, then `year` is included as a random factor. If set to `FALSE`, then `year` is included as a fixed factor. All other combinations of logical value and `approach` lead to `year` not being included in modeling. Defaults to `TRUE`.

`juvestimate` An optional variable denoting the stage name of the juvenile stage in the vertical dataset. If not `NA`, and `stage` is also given (see below), then vital rates listed in `vitalrates` other than `fec` will also be estimated from the juvenile stage to all adult stages. Defaults to `NA`, in which case juvenile vital rates are not estimated.

`juvsize` A logical variable denoting whether size should be used as a term in models involving transition from the juvenile stage. Defaults to `FALSE`, and is only used if `juvestimate` does not equal `NA`.

`fectime` A variable indicating which year of fecundity to use as the response term in fecundity models. Options include `2`, which refers to occasion *t*, and `3`, which refers to occasion *t*+1. Defaults to `2`.

`censor` A vector denoting the names of censoring variables in the dataset, in order from occasion *t*+1, followed by occasion *t*, and lastly followed by occasion *t*-1. Defaults to `NA`.

`age` Designates the name of the variable corresponding to age in time *t* in the vertical dataset. Defaults to `NA`, in which case age is not included in linear models. Should only be used if building Leslie or age x stage matrices.

`indcova` Vector designating the names in occasions *t*+1, *t*, and *t*-1 of an individual covariate. Defaults to `NA`.

`indcovb` Vector designating the names in occasions *t*+1, *t*, and *t*-1 of a second individual covariate. Defaults to `NA`.

`indcovc` Vector designating the names in occasions *t*+1, *t*, and *t*-1 of a third individual covariate. Defaults to `NA`.

random.indcova	A logical value indicating whether indcova should be treated as a random categorical factor, rather than as a fixed factor. Defaults to FALSE.
random.indcovb	A logical value indicating whether indcovb should be treated as a random categorical factor, rather than as a fixed factor. Defaults to FALSE.
random.indcovc	A logical value indicating whether indcovc should be treated as a random categorical factor, rather than as a fixed factor. Defaults to FALSE.
test.group	A logical value indicating whether to include the group variable from the input stageframe as a fixed categorical variable in linear models. Defaults to FALSE.
...	Other parameters.

Value

This function yields text output describing the subsets to be used in linear vital rate modeling. No value or object is returned.

Notes

This function is meant to handle input as would be supplied to function modelsearch(). To use most easily, users may copy all input parameters from a call to function modelsearch(), and paste directly within this function. The exact subsets used in the modelsearch() run will also be created here.

Tests of Gaussian normality are conducted as Shapiro-Wilk tests via base R's shapiro.test() function. If datasets with more than 5000 rows are supplied, function hfv_qc() will sample 5000 rows from the dataset and conduct the Shapiro-Wilk test on the data sample.

Random factor variables are also tested for the presence of singleton categories, which are factor values that occur only once in the used data subset. Singleton categories may cause problems with estimation under mixed modeling.

Examples

```
data(lathyrus)

sizevector <- c(0, 4.6, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 1, 2, 3, 4, 5, 6, 7, 8,
  9)
stagevector <- c("Sd", "Sdl", "Dorm", "Sz1nr", "Sz2nr", "Sz3nr", "Sz4nr",
  "Sz5nr", "Sz6nr", "Sz7nr", "Sz8nr", "Sz9nr", "Sz1r", "Sz2r", "Sz3r",
  "Sz4r", "Sz5r", "Sz6r", "Sz7r", "Sz8r", "Sz9r")
repvector <- c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1)
obsvector <- c(0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
matvector <- c(0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0)
indataset <- c(0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 4.6, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
  0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5)

lathframeln <- sf_create(sizes = sizevector, stagenames = stagevector,
  repstatus = repvector, obsstatus = obsvector, matstatus = matvector,
```

```
  immstatus = immvector, indataset = indataset, binhalfwidth = binvec,
  propstatus = propvector)

lathvertln <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "lnVol88", repstracol = "Intactseed88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframeln, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, NAas0 = TRUE, censor = TRUE)

lathvertln$feca2 <- round(lathvertln$feca2)
lathvertln$feca1 <- round(lathvertln$feca1)
lathvertln$feca3 <- round(lathvertln$feca3)

hfv_qc(lathvertln, historical = TRUE, suite = "main",
  vitalrates = c("surv", "obs", "size", "repst", "fec"), juvestimate = "Sdl",
  indiv = "individ", patch = "patchid", year = "year2",year.as.random = TRUE,
  patch.as.random = TRUE)
```

`historicalize3` *Create Historical Vertical Data Frame from Ahistorical Vertical Data Frame*

Description

Function `historicalize3()` returns a vertically formatted demographic data frame organized to create historical projection matrices, given a vertically but ahistorically formatted data frame. This data frame is in standard `hfvdata` format and can be used in all functions in the package.

Usage

```
historicalize3(
  data,
  popidcol = 0,
  patchidcol = 0,
  individcol,
  year2col = 0,
  year3col = 0,
  xcol = 0,
  ycol = 0,
  sizea2col = 0,
  sizea3col = 0,
  sizeb2col = 0,
  sizeb3col = 0,
  sizec2col = 0,
  sizec3col = 0,
  repstra2col = 0,
  repstra3col = 0,
```

```
  repstrb2col = 0,
  repstrb3col = 0,
  feca2col = 0,
  feca3col = 0,
  fecb2col = 0,
  fecb3col = 0,
  indcova2col = 0,
  indcova3col = 0,
  indcovb2col = 0,
  indcovb3col = 0,
  indcovc2col = 0,
  indcovc3col = 0,
  alive2col = 0,
  alive3col = 0,
  dead2col = 0,
  dead3col = 0,
  obs2col = 0,
  obs3col = 0,
  nonobs2col = 0,
  nonobs3col = 0,
  repstrrel = 1,
  fecrel = 1,
  stage2col = 0,
  stage3col = 0,
  juv2col = 0,
  juv3col = 0,
  stageassign = NA,
  stagesize = NA,
  censor = FALSE,
  censorcol = 0,
  censorkeep = 0,
  spacing = NA,
  NAas0 = FALSE,
  NRasRep = FALSE,
  NOasObs = FALSE,
  prebreeding = TRUE,
  age_offset = 0,
  reduce = TRUE,
  a2check = FALSE,
  quiet = FALSE
)
```

Arguments

data	The horizontal data file.
popidcol	A variable name or column number corresponding to the identity of the population for each individual.
patchidcol	A variable name or column number corresponding to the identity of the patch

or subpopulation for each individual, if patches have been designated within populations.

`individcol` A variable name or column number corresponding to the unique identity of each individual.

`year2col` A variable name or column number corresponding to occasion *t* (year or time).

`year3col` A variable name or column number corresponding to occasion *t*+1 (year or time).

`xcol` A variable name or column number corresponding to the X coordinate of each individual in Cartesian space.

`ycol` A variable name or column number corresponding to the Y coordinate of each individual in Cartesian space.

`sizea2col` A variable name or column number corresponding to the primary size entry in occasion *t*.

`sizea3col` A variable name or column number corresponding to the primary size entry in occasion *t*+1.

`sizeb2col` A variable name or column number corresponding to the secondary size entry in occasion *t*.

`sizeb3col` A variable name or column number corresponding to the secondary size entry in occasion *t*+1.

`sizec2col` A variable name or column number corresponding to the tertiary size entry in occasion *t*.

`sizec3col` A variable name or column number corresponding to the tertiary size entry in occasion *t*+1.

`repstra2col` A variable name or column number corresponding to the production of reproductive structures, such as flowers, in occasion *t*. This can be binomial or count data, and is used to in analysis of the probability of reproduction.

`repstra3col` A variable name or column number corresponding to the production of reproductive structures, such as flowers, in occasion *t*+1. This can be binomial or count data, and is used to in analysis of the probability of reproduction.

`repstrb2col` A second variable name or column number corresponding to the production of reproductive structures, such as flowers, in occasion *t*. This can be binomial or count data.

`repstrb3col` A second variable name or column number corresponding to the production of reproductive structures, such as flowers, in occasion *t*+1. This can be binomial or count data.

`feca2col` A variable name or column number corresponding to fecundity in occasion *t*. This may represent egg counts, fruit counts, seed production, etc.

`feca3col` A variable name or column number corresponding to fecundity in occasion *t*+1. This may represent egg counts, fruit counts, seed production, etc.

`fecb2col` A second variable name or column number corresponding to fecundity in occasion *t*. This may represent egg counts, fruit counts, seed production, etc.

`fecb3col` A second variable name or column number corresponding to fecundity in occasion *t*+1. This may represent egg counts, fruit counts, seed production, etc.

`indcova2col` A variable name or column number corresponding to an individual covariate to be used in analysis, in occasion *t*.

`indcova3col` A variable name or column number corresponding to an individual covariate to be used in analysis, in occasion *t*+1.

`indcovb2col` A variable name or column number corresponding to a second individual covariate to be used in analysis, in occasion *t*.

`indcovb3col` A variable name or column number corresponding to a second individual covariate to be used in analysis, in occasion *t*+1.

`indcovc2col` A variable name or column number corresponding to a third individual covariate to be used in analysis, in occasion *t*.

`indcovc3col` A variable name or column number corresponding to a third individual covariate to be used in analysis, in occasion *t*+1.

`alive2col` A variable name or column number that provides information on whether an individual is alive in occasion *t*. If used, living status must be designated as binomial (living = `1`, dead = `0`).

`alive3col` A variable name or column number that provides information on whether an individual is alive in occasion *t*+1. If used, living status must be designated as binomial (living = `1`, dead = `0`).

`dead2col` A variable name or column number that provides information on whether an individual is dead in occasion *t*. If used, dead status must be designated as binomial (living = `0`, dead = `1`).

`dead3col` A variable name or column number that provides information on whether an individual is dead in occasion *t*+1. If used, dead status must be designated as binomial (living = `0`, dead = `1`).

`obs2col` A variable name or column number providing information on whether an individual is in an observable stage in occasion *t*. If used, observation status must be designated as binomial (observed = `1`, not observed = `0`).

`obs3col` A variable name or column number providing information on whether an individual is in an observable stage in occasion *t*+1. If used, observation status must be designated as binomial (observed = `1`, not observed = `0`).

`nonobs2col` A variable name or column number providing information on whether an individual is in an unobservable stage in occasion *t*. If used, observation status must be designated as binomial (observed = `0`, not observed = `1`).

`nonobs3col` A variable name or column number providing information on whether an individual is in an unobservable stage in occasion *t*+1. If used, observation status must be designated as binomial (observed = `0`, not observed = `1`).

`repstrrel` This is a scalar multiplier making the variable represented by `repstrb2col` equivalent to the variable represented by `repstra2col`. This can be useful if two reproductive status variables have related but unequal units, for example if `repstrb2col` refers to one-flowered stems while `repstra2col` refers to two-flowered stems.

`fecrel` This is a scalar multiplier making the variable represented by `fecb2col` equivalent to the variable represented by `feca2col`. This can be useful if two fecundity variables have related but unequal units.

stage2col — Optional variable name or column number corresponding to life history stage in occasion *t*.

stage3col — Optional variable name or column number corresponding to life history stage in occasion *t*+1.

juv2col — A variable name or column number that marks individuals in immature stages in occasion *t*. Function historicalize3() assumes that immature individuals are identified in this variable marked with a number equal to or greater than 1, and that mature individuals are marked as 0 or NA.

juv3col — A variable name or column number that marks individuals in immature stages in occasion *t*+1. Function historicalize3() assumes that immature individuals are identified in this variable marked with a number equal to or greater than 1, and that mature individuals are marked as 0 or NA.

stageassign — The stageframe object identifying the life history model being operationalized. Note that if stage2col is provided, then this stageframe is not utilized in stage designation.

stagesize — A variable name or column number describing which size variable to use in stage estimation. Defaults to NA, and can also take sizea, sizeb, sizec, sizeab, sizebc, sizeac, sizeabc, or sizeadded, depending on which size variable within the input dataset is chosen. Note that the variable(s) chosen should be presented in the order of the primary, secondary, and tertiary variables in the stageframe input with stageassign. For example, choosing sizeb assumes that this size variable in the dataset is the primary variable in the stageframe.

censor — A logical variable determining whether the output data should be censored using the variable defined in censorcol. Defaults to FALSE.

censorcol — A variable name or column number corresponding to a censor variable within the dataset, used to distinguish between entries to use and those to discard from analysis, or to designate entries with special issues that require further attention.

censorkeep — The value of the censoring variable identifying data that should be included in analysis. Defaults to 0, but may take any value including NA.

spacing — The spacing at which density should be estimated, if density estimation is desired and X and Y coordinates are supplied. Given in the same units as those used in the X and Y coordinates given in xcol and ycol. Defaults to NA.

NAas0 — If TRUE, then all NA entries for size and fecundity variables will be set to 0. This can help increase the sample size analyzed by modelsearch(), but should only be used when it is clear that this substitution is biologically realistic. Defaults to FALSE.

NRasRep — If set to TRUE, then this function will treat non-reproductive but mature individuals as reproductive during stage assignment. This can be useful when a matrix is desired without separation of reproductive and non-reproductive but mature stages of the same size. Only used if stageassign is set to a valid stageframe. Defaults to FALSE.

NOasObs — If TRUE, then will treat individuals that are interpreted as not observed in the dataset as though they were observed during stage assignment. This can be useful when a MPM is desired without separation of observable and unobservable stages. Only used if stageassign is set to a stageframe. Defaults to FALSE.

prebreeding	A logical term indicating whether the life history model is pre-breeding. If so, then 1 is added to all ages. Defaults to TRUE.
age_offset	A number to add automatically to all values of age at time *t*. Defaults to 0.
reduce	A logical variable determining whether unused variables and some invariant state variables should be removed from the output dataset. Defaults to TRUE.
a2check	A logical variable indicating whether to retain all data with living status at occasion *t*. Defaults to FALSE, in which case data for occasions in which the individual is not alive in time *t* is not retained. This option should be kept FALSE, except to inspect potential errors in the dataset.
quiet	A logical variable indicating whether to silence warnings. Defaults to FALSE.

Value

If all inputs are properly formatted, then this function will output a historical vertical data frame (class hfvdata), meaning that the output data frame will have three consecutive years of size and reproductive data per individual per row. This data frame is in standard format for all functions used in lefko3, and so can be used without further modification. Note that determination of state in occasions *t*-1 and *t*+1 gives preference to condition in occasion *t* within the input dataset. Conflicts in condition in input datasets that have both occasions *t* and *t*+1 listed per row are resolved by using condition in occasion *t*.

Variables in this data frame include the following:

rowid	Unique identifier for the row of the data frame.
popid	Unique identifier for the population, if given.
patchid	Unique identifier for patch within population, if given.
individ	Unique identifier for the individual.
year2	Year or time in occasion *t*.
firstseen	Occasion of first observation.
lastseen	Occasion of last observation.
obsage	Observed age in occasion *t*, assuming first observation corresponds to age = 0.
obslifespan	Observed lifespan, given as lastseen - firstseen + 1.
xpos1, xpos2, xpos3	X position in Cartesian space in occasions *t*-1, *t*, and *t*+1, respectively, if provided.
ypos1, ypos2, ypos3	Y position in Cartesian space in occasions *t*-1, *t*, and *t*+1, respectively, if provided.
sizea1, sizea2, sizea3	Main size measurement in occasions *t*-1, *t*, and *t*+1, respectively.
sizeb1, sizeb2, sizeb3	Secondary size measurement in occasions *t*-1, *t*, and *t*+1, respectively.
sizec1, sizec2, sizec3	Tertiary size measurement in occasions *t*-1, *t*, and *t*+1, respectively.

size1added, size2added, size3added
: Sum of primary, secondary, and tertiary size measurements in occasions *t*-1, *t*, and *t*+1, respectively.

repstra1, repstra2, repstra3
: Main numbers of reproductive structures in occasions *t*-1, *t*, and *t*+1, respectively.

repstrb1, repstrb2, repstrb3
: Secondary numbers of reproductive structures in occasions *t*-1, *t*, and *t*+1, respectively.

repstr1added, repstr2added, repstr3added
: Sum of primary and secondary reproductive structures in occasions *t*-1, *t*, and *t*+1, respectively.

feca1, feca2, feca3
: Main numbers of offspring in occasions *t*-1, *t*, and *t*+1, respectively.

fecb1, fecb2, fecb3
: Secondary numbers of offspring in occasions *t*-1, *t*, and *t*+1, respectively.

fec1added, fec2added, fec3added
: Sum of primary and secondary fecundity in occasions *t*-1, *t*, and *t*+1, respectively.

censor1, censor2, censor3
: Censor status values in occasions *t*-1, *t*, and *t*+1, respectively.

juvgiven1, juvgiven2, juvgiven3
: Binomial variable indicating whether individual is juvenile in occasions *t*-1, *t*, and *t*+1. Only given if juvcol is provided.

obsstatus1, obsstatus2, obsstatus3
: Binomial observation status in occasions *t*-1, *t*, and *t*+1, respectively.

repstatus1, repstatus2, repstatus3
: Binomial reproductive status in occasions *t*-1, *t*, and *t*+1, respectively.

fecstatus1, fecstatus2, fecstatus3
: Binomial offspring production status in occasions *t*-1, *t*, and *t*+1, respectively.

matstatus1, matstatus2, matstatus3
: Binomial maturity status in occasions *t*-1, *t*, and *t*+1, respectively.

alive1, alive2, alive3
: Binomial status as alive in occasions *t*-1, *t*, and *t*+1, respectively.

density
: Density of individuals per unit designated in spacing. Only given if spacing is not NA.

Notes

Warnings that some individuals occur in state combinations that do not match any stages in the stageframe used to assign stages, and that some individuals match characteristics of several stages in the stageframe, are common when first working with a dataset. Typically, these situations can be identified as NoMatch entries in stage3, although such entries may crop up in stage1 and stage2, as well. In some cases, these warnings will arise with no concurrent NoMatch entries. These are important warnings and suggest that there is likely a problem with the stageframe. The most common such problems are: 1) stages have significant overlap in characteristics, with the most common

being overlapping size bins caused by erroneous definitions of size bin halfwidths; and 2) some individuals exist in states not defined within the stageframe.

In some datasets with unobservable stages, observation status (obsstatus) might not be inferred properly if a single size variable is used that does not yield sizes greater than 0 in all cases in which individuals were observed. Such situations may arise, for example, in plants when leaf number is the dominant size variable used, but individuals occasionally occur with inflorescences but no leaves. In this instances, it helps to mark related variables as sizeb and sizec, because observation status will be interpreted in relation to all 3 size variables. Alternatively, observation status may be input via obs2col and obs3col to force computation with given values (although this requires all instances of observation and non-observation to be known and coded ahead of time). Further analysis can then utilize only a single size variable, of the user's choosing. Similar issues can arise in reproductive status (repstatus).

Juvenile designation should only be used when juveniles fall outside of the size classification scheme used in determining stages. If juveniles are to be size classified along the size spectrum that adults also fall on, then it is best to treat juveniles as mature but not reproductive.

Care should be taken to avoid variables with negative values indicating size, fecundity, or reproductive or observation status. Negative values can be interpreted in different ways, typically reflecting estimation through other algorithms rather than actual measured data. Variables holding negative values can conflict with data management algorithms in ways that are difficult to predict.

Unusual errors (e.g. "Error in pjf...") may occur in cases where the variables are improperly passed, or where seemingly numeric variables include text and so get automatically converted to string variables.

Density estimation is performed as a count of individuals alive and within the radius specified in spacing of the respective individual at some point in time.

Examples

```
data(cypvert)

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_v2 <- historicalize3(data = cypvert, patchidcol = "patch",
  individcol = "plantid", year2col = "year2", sizea2col = "Inf2.2",
  sizea3col = "Inf2.3", sizeb2col = "Inf.2", sizeb3col = "Inf.3",
  sizec2col = "Veg.2", sizec3col = "Veg.3", repstra2col = "Inf2.2",
```

```
repstra3col = "Inf2.3", repstrb2col = "Inf.2", repstrb3col = "Inf.3",
feca2col = "Pod.2", feca3col = "Pod.3", repstrrel = 2,
stageassign = cypframe_raw, stagesize = "sizeadded", censorcol = "censor",
censor = FALSE, NAas0 = TRUE, NRasRep = TRUE, reduce = TRUE)
```

`hist_null` *Create Historical MPMs Assuming No Influence of Individual History*

Description

Function `hist_null()` uses ahistorical MPMs to create the equivalent MPMs in the structure of historical MPMs. These MPMs have the same dimensions and stage structure of hMPMs but assume no influence of individual history, and so can be compared to actual hMPMs.

Usage

```
hist_null(mpm, format = 1L, err_check = FALSE)
```

Arguments

`mpm`	An ahistorical MPM of class `lefkoMat`.
`format`	An integer stipulating whether historical matrices should be produced in Ehrlen format (`1`) or deVries format (`2`).
`err_check`	A logical value indicating whether to output the main index data frames used to sort elements in the matrices.

Value

An object of class `lefkoMat`, with the same list structure as the input object, but with `A`, `U`, and `F` elements replaced with lists of historically-structured matrices, and with element `hstages` changed from `NA` to an index of stage pairs corresponding to the rows and columns of the new matrices. If `err_check = TRUE`, then a list of three data frames showing the values used to determine matrix element index values is also exported.

Notes

This function does not currently identify biologically impossible transitions. Ahistorical transition values are placed in all theoretically possible historical transitions.

Examples

```
sizevector <- c(1, 1, 2, 3)
stagevector <- c("Sdl", "Veg", "SmFlo", "LFlo")
repvector <- c(0, 0, 1, 1)
obsvector <- c(1, 1, 1, 1)
matvector <- c(0, 1, 1, 1)
immvector <- c(1, 0, 0, 0)
```

```
propvector <- c(0, 0, 0, 0)
indataset <- c(1, 1, 1, 1)
binvec <- c(0.5, 0.5, 0.5, 0.5)

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

# POPN C 2003-2004
XC3 <- matrix(c(0, 0, 1.74, 1.74,
0.208333333, 0, 0, 0.057142857,
0.041666667, 0.076923077, 0, 0,
0.083333333, 0.076923077, 0.066666667, 0.028571429), 4, 4, byrow = TRUE)

# 2004-2005
XC4 <- matrix(c(0, 0, 0.3, 0.6,
0.32183908, 0.142857143, 0, 0,
0.16091954, 0.285714286, 0, 0,
0.252873563, 0.285714286, 0.5, 0.6), 4, 4, byrow = TRUE)

mats_list <- list(XC3, XC4)
yr_ord <- c(1, 2)
pch_ord <- c(1, 1)

anth_lefkoMat <- create_lM(mats_list, anthframe, hstages = NA, historical = FALSE,
  poporder = 1, patchorder = pch_ord, yearorder = yr_ord)

nullmodel1 <- hist_null(anth_lefkoMat, 1) # Ehrlen format
nullmodel2 <- hist_null(anth_lefkoMat, 2) # deVries format
```

`image3` *Create Matrix Image*

Description

Function `image3()` is a generic function that creates matrix plots.

Usage

```
image3(mats, ...)
```

Arguments

`mats`	A lefkoMat object, or a single projection matrix, for which the dominant eigenvalue is desired.
`...`	Other parameters

Value

Produces a single matrix image, or a series of images, depending on the input. Non-zero elements appear as red space, while zero elements appear as white space.

See Also

image3.lefkoMat()

image3.matrix()

Examples

```
# Lathyrus example
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
matvector <- c(0, 0, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)

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

lathvert <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "Volume88", repstracol = "FCODE88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframe, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, censor = TRUE)

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "Sd", "Sdl", "mat"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "rep", "rep", "Sdl"),
  stage1 = c("Sd", "rep", "Sd", "rep", "npr", "npr", "Sd"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, "mat"),
  eststage2 = c(NA, NA, NA, NA, NA, NA, "Sdl"),
  eststage1 = c(NA, NA, NA, NA, NA, NA, "NotAlive"),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, 0.345, 0.054, NA),
  type = c(1, 1, 1, 1, 3, 3, 1), type_t12 = c(1, 2, 1, 2, 1, 1, 1),
  stageframe = lathframe, historical = TRUE)

ehrlen3 <- rlefko3(data = lathvert, stageframe = lathframe, year = "all",
  stages = c("stage3", "stage2", "stage1"), supplement = lathsupp3,
  yearcol = "year2", indivcol = "individ")
```

```
image3(ehrlen3, used = 1, type = "U")
```

`image3.dgCMatrix` *Create a Matrix Image for a Single Sparse Matrix*

Description

Function `image3.dgCMatrix` plots a matrix image for a single sparse matrix.

Usage

```
## S3 method for class 'dgCMatrix'
image3(mats, ...)
```

Arguments

`mats`	A `matrix` class object.
`...`	Other parameters.

Value

Plots a matrix image, or series of matrix images, denoting non-zero elements as red space and zero elements as white space.

Examples

```
# Lathyrus example
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
matvector <- c(0, 0, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)

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

lathvert <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "Volume88", repstracol = "FCODE88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
```

```
  nonobsacol = "Dormant1988", stageassign = lathframe, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, censor = TRUE)

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "Sd", "Sdl", "mat"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "rep", "rep", "Sdl"),
  stage1 = c("Sd", "rep", "Sd", "rep", "npr", "npr", "Sd"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, "mat"),
  eststage2 = c(NA, NA, NA, NA, NA, NA, "Sdl"),
  eststage1 = c(NA, NA, NA, NA, NA, NA, "NotAlive"),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, 0.345, 0.054, NA),
  type = c(1, 1, 1, 1, 3, 3, 1), type_t12 = c(1, 2, 1, 2, 1, 1, 1),
  stageframe = lathframe, historical = TRUE)

ehrlen3 <- rlefko3(data = lathvert, stageframe = lathframe, year = "all",
  stages = c("stage3", "stage2", "stage1"), supplement = lathsupp3,
  yearcol = "year2", indivcol = "individ", sparse_output = TRUE)

image3(ehrlen3$U[[1]])
```

`image3.lefkoElas` *Create Matrix Image(s) for lefkoElas Object*

Description

Function `image3.lefkoElas` plots matrix images for elasticity matrices supplied within `lefkoElas` objects.

Usage

```
## S3 method for class 'lefkoElas'
image3(mats, used = "all", type = "a", ...)
```

Arguments

`mats`	A `lefkoElas` object.
`used`	A numeric value or vector designating the matrices to plot. Can also take the value `"all"`, which plots all matrices. Defaults to `"all"`.
`type`	Character value indicating whether to plot `"a"`historical or `"h"`istorical elasticity matrices. Defaults to `"a"`historical, but will plot a historical elasticity matrix image if no ahistorical elasticity matrix exists.
`...`	Other parameters.

Value

Plots a matrix image, or series of matrix images, denoting non-zero elements as red space and zero elements as white space.

Examples

```
# Lathyrus example
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
matvector <- c(0, 0, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)

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

lathvert <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "Volume88", repstracol = "FCODE88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframe, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, censor = TRUE)

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "Sd", "Sdl", "mat"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "rep", "rep", "Sdl"),
  stage1 = c("Sd", "rep", "Sd", "rep", "npr", "npr", "Sd"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, "mat"),
  eststage2 = c(NA, NA, NA, NA, NA, NA, "Sdl"),
  eststage1 = c(NA, NA, NA, NA, NA, NA, "NotAlive"),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, 0.345, 0.054, NA),
  type = c(1, 1, 1, 1, 3, 3, 1), type_t12 = c(1, 2, 1, 2, 1, 1, 1),
  stageframe = lathframe, historical = TRUE)

ehrlen3 <- rlefko3(data = lathvert, stageframe = lathframe,
  year = c(1989, 1990), stages = c("stage3", "stage2", "stage1"),
  supplement = lathsupp3, yearcol = "year2", indivcol = "individ")

ehrlen_elas <- elasticity3(ehrlen3)

image3(ehrlen_elas, used = 1, type = "h")
```

`image3.lefkoMat` *Create Matrix Image(s) for lefkoMat Object*

Description

Function `image3.lefkoMat` plots matrix images for matrices supplied within `lefkoMat` objects.

Usage

```
## S3 method for class 'lefkoMat'
image3(mats, used = "all", type = "A", ...)
```

Arguments

`mats`	A `lefkoMat` object.
`used`	A numeric value or vector designating the matrices to plot. Can also take the value `"all"`, which plots all matrices. Defaults to `"all"`.
`type`	Character value indicating whether to plot `A`, `U`, or `F` matrices. Defaults to `"A"`.
`...`	Other parameters.

Value

Plots a matrix image, or series of matrix images, denoting non-zero elements as red space and zero elements as white space.

Examples

```
# Lathyrus example
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
matvector <- c(0, 0, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)

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

lathvert <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "Volume88", repstracol = "FCODE88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframe, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, censor = TRUE)

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "Sd", "Sdl", "mat"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "rep", "rep", "Sdl"),
```

```
  stage1 = c("Sd", "rep", "Sd", "rep", "npr", "npr", "Sd"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, "mat"),
  eststage2 = c(NA, NA, NA, NA, NA, NA, "Sdl"),
  eststage1 = c(NA, NA, NA, NA, NA, NA, "NotAlive"),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, 0.345, 0.054, NA),
  type = c(1, 1, 1, 1, 3, 3, 1), type_t12 = c(1, 2, 1, 2, 1, 1, 1),
  stageframe = lathframe, historical = TRUE)

ehrlen3 <- rlefko3(data = lathvert, stageframe = lathframe, year = "all",
  stages = c("stage3", "stage2", "stage1"), supplement = lathsupp3,
  yearcol = "year2", indivcol = "individ")

image3(ehrlen3, used = 1, type = "U")
```

`image3.lefkoSens` *Create Matrix Image(s) for lefkoSens Object*

Description

Function `image3.lefkoSens` plots matrix images for sensitivity matrices supplied within `lefkoSens` objects.

Usage

```
## S3 method for class 'lefkoSens'
image3(mats, used = "all", type = "a", ...)
```

Arguments

`mats`	A `lefkoSens` object.
`used`	A numeric value or vector designating the matrices to plot. Can also take the value `"all"`, which plots all matrices. Defaults to `"all"`.
`type`	Character value indicating whether to plot `"a"`historical or `"h"`istorical sensitivity matrices. Defaults to `"a"`historical, but will plot a historical sensitivity matrix image if no ahistorical sensitivity matrix exists.
`...`	Other parameters.

Value

Plots a matrix image, or series of matrix images, denoting non-zero elements as red space and zero elements as white space.

Examples

```
# Lathyrus example
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
matvector <- c(0, 0, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)

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

lathvert <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "Volume88", repstracol = "FCODE88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframe, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, censor = TRUE)

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "Sd", "Sdl", "mat"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "rep", "rep", "Sdl"),
  stage1 = c("Sd", "rep", "Sd", "rep", "npr", "npr", "Sd"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, "mat"),
  eststage2 = c(NA, NA, NA, NA, NA, NA, "Sdl"),
  eststage1 = c(NA, NA, NA, NA, NA, NA, "NotAlive"),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, 0.345, 0.054, NA),
  type = c(1, 1, 1, 1, 3, 3, 1), type_t12 = c(1, 2, 1, 2, 1, 1, 1),
  stageframe = lathframe, historical = TRUE)

ehrlen3 <- rlefko3(data = lathvert, stageframe = lathframe,
  year = c(1989, 1990), stages = c("stage3", "stage2", "stage1"),
  supplement = lathsupp3, yearcol = "year2", indivcol = "individ")

ehrlen_sens <- sensitivity3(ehrlen3)

image3(ehrlen_sens, used = 1, type = "h")
```

`image3.list` *Create Matrix Images for Matrices in a List*

Description

Function `image3.list` plots matrix images for matrices contained in a list of matrices.

Usage

```
## S3 method for class 'list'
image3(mats, used = "all", ...)
```

Arguments

`mats`	A `list` class object.
`used`	A numeric vector of projection matrices within `mats` to represent as matrix images. Can also take the text value `"all"`, which will produce images of all matrices. Defaults to `"all"`.
`...`	Other parameters.

Value

Plots a matrix image, or series of matrix images, denoting non-zero elements as red space and zero elements as white space.

Examples

```
# Lathyrus example
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
matvector <- c(0, 0, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)

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

lathvert <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "Volume88", repstracol = "FCODE88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframe, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, censor = TRUE)

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "Sd", "Sdl"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "rep", "rep"),
  stage1 = c("Sd", "rep", "Sd", "rep", "all", "all"),
```

```
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA),
  multiplier = c(NA, NA, NA, NA, 0.345, 0.054),
  type = c(1, 1, 1, 1, 3, 3), type_t12 = c(1, 2, 1, 2, 1, 1),
  stageframe = lathframe, historical = TRUE)

ehrlen3 <- rlefko3(data = lathvert, stageframe = lathframe, year = "all",
  stages = c("stage3", "stage2", "stage1"), supplement = lathsupp3,
  yearcol = "year2", indivcol = "individ")

image3(ehrlen3$A, used = 1)
```

`image3.matrix` *Create a Matrix Image for a Single Matrix*

Description

Function `image3.matrix` plots a matrix image for a single matrix.

Usage

```
## S3 method for class 'matrix'
image3(mats, ...)
```

Arguments

`mats`	A `matrix` class object.
`...`	Other parameters.

Value

Plots a matrix image, or series of matrix images, denoting non-zero elements as red space and zero elements as white space.

Examples

```
# Lathyrus example
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
matvector <- c(0, 0, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)
```

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

lathvert <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "Volume88", repstracol = "FCODE88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframe, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, censor = TRUE)

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "Sd", "Sdl", "mat"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "rep", "rep", "Sdl"),
  stage1 = c("Sd", "rep", "Sd", "rep", "npr", "npr", "Sd"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, "mat"),
  eststage2 = c(NA, NA, NA, NA, NA, NA, "Sdl"),
  eststage1 = c(NA, NA, NA, NA, NA, NA, "NotAlive"),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, 0.345, 0.054, NA),
  type = c(1, 1, 1, 1, 3, 3, 1), type_t12 = c(1, 2, 1, 2, 1, 1, 1),
  stageframe = lathframe, historical = TRUE)

ehrlen3 <- rlefko3(data = lathvert, stageframe = lathframe, year = "all",
  stages = c("stage3", "stage2", "stage1"), supplement = lathsupp3,
  yearcol = "year2", indivcol = "individ", sparse_output = FALSE)

image3(ehrlen3$U[[1]])
```

`lambda3` *Estimate Actual or Deterministic Population Growth Rate*

Description

Function `lambda3()` is a generic function that returns the dominant eigenvalue of a matrix, set of dominant eigenvalues of a set of matrices, set of dominant eigenvalues for a `lefkoMat` object, or actual λ in each year in a `lefkoProj` object. It can handle large and sparse matrices supplied as `lefkoMat` objects or as individual matrices, and can be used with large historical matrices, IPMs, age x stage matrices, as well as smaller ahistorical matrices, and general projetions.

Usage

```
lambda3(mpm, style = NULL, force_sparse = NULL)
```

Arguments

`mpm`	A `lefkoMat` object, a `lefkoMatList` object, a list of projection matrices, a `lefkoProj` object, or a single projection matrix in either standard or sparse format.

style — If mpm is either a lefkoMat or lefkoMatList object, then style determines whether the dominant eigenvalues will be calculated for the A matrices (the default), the U matrices (if style = "U"), or the F matrices (if style = "F").

force_sparse — A logical value or string detailing whether to force sparse matrix encoding for simple matrix input. Defaults to "auto", which only forces sparse matrix coding if simple matrices are input that are both sparse (i.e, percentage of matrix elements that are non-zero <= 50 and have more than 20 rows. Can also be set to "yes", "no", TRUE, or FALSE. Note that sparse matrix coding is always used for lefkoMat objects with matrices in sparse format (class dgCMatrix). Ignored with lefkoProj objects.

Value

The value returned depends on the class of the mpm argument. If a lefkoMat object is provided, then this function will return the labels data frame with a new column named lambda showing the dominant eigenvalues for each matrix. If a lefkoMatList object is provided, then a two element list in which the first element is a vector composed of the mean lambda values of each lefkoMat element within the list is provided, and the second element is a list of lefkoMat lambda summaries as previously described. If a list of matrices is provided, then this function will produce a numeric vector with the dominant eigenvalues provided in order of matrix. If a single matrix is provided, then this function will return the dominant eigenvalue of that matrix. Only the largest real parts of the eigenvalues are returned.

If a lefkoProj object is provided, then the output consists of a list with three elements. The second and third elements are lists of matrices with each lower-level list elements corresponding to labels rows, and matrices within these lists showing the actual λ and log λ for each consecutive year or time index (columns) within each replicate (row).

See Also

slambda3()

Examples

```
# Lathyrus example
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
matvector <- c(0, 0, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)

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

```
lathvert <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "Volume88", repstracol = "FCODE88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframe, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, censor = TRUE)

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "Sd", "Sdl", "mat"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "rep", "rep", "Sdl"),
  stage1 = c("Sd", "rep", "Sd", "rep", "npr", "npr", "Sd"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, "mat"),
  eststage2 = c(NA, NA, NA, NA, NA, NA, "Sdl"),
  eststage1 = c(NA, NA, NA, NA, NA, NA, "NotAlive"),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, 0.345, 0.054, NA),
  type = c(1, 1, 1, 1, 3, 3, 1), type_t12 = c(1, 2, 1, 2, 1, 1, 1),
  stageframe = lathframe, historical = TRUE)

ehrlen3 <- rlefko3(data = lathvert, stageframe = lathframe, year = "all",
  stages = c("stage3", "stage2", "stage1"), supplement = lathsupp3,
  yearcol = "year2", indivcol = "individ")

ehrlen3mean <- lmean(ehrlen3)
lambda3(ehrlen3mean)

# Cypripedium example
data(cypdata)

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 = 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)

# Here we use supplemental() to provide overwrite and reproductive info
```

```
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", patch = "all", stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added"), supplement = cypsupp2r,
  yearcol = "year2", patchcol = "patchid", indivcol = "individ")

lambda3(cypmatrix2r)
```

`lathyrus` *Demographic Dataset of* Lathyrus vernus *Population*

Description

A dataset containing the states and fates of *Lathyrus vernus* (spring vetch), family Fabaceae, from a population in Sweden monitored annually from 1988 to 1991 in six study plots.

Usage

```
data(lathyrus)
```

Format

A data frame with 1119 individuals and 34 variables. Each row corresponds to a unique individual, and each variable from `Volume88` on refers to the state of the individual in a given year.

SUBPLOT A variable refering to patch within the population.

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

Volume88 Aboveground volume in cubic mm in 1988.

lnVol88 Natural logarithm of `Volume88`.

FCODE88 Equals 1 if flowering and 0 if not flowering in 1988.

Flow88 Number of flowers in 1988.

Intactseed88 Number of intact mature seeds produced in 1988. Not always an integer, as in some cases seed number was estimated via linear modeling.

Dead1988 Marked as 1 if known to be dead in 1988.

Dormant1988 Marked as 1 if known to be alive but vegetatively dormant in 1988.

Missing1988 Marked as 1 if not found in 1988.

Seedling1988 Marked as 1, 2, or 3 if observed as a seedling in year *t*. Numbers refer to certainty of assignment: 1 = certain that plant is a seedling in 1988, 2 = likely that plant is a seedling in 1988, 3 = probable that plant is a seedling in 1988.

Volume89 Aboveground volume in cubic mm in 1989.

lnVol89 Natural logarithm of `Volume89`.

FCODE89 Equals 1 if flowering and 0 if not flowering in 1989.

Flow89 Number of flowers in 1989.

Intactseed89 NZumber of intact mature seeds produced in 1989. Not always an integer, as in some cases seed number was estimated via linear modeling.

Dead1989 Marked as 1 if known to be dead in 1989.

Dormant1989 Marked as 1 if known to be alive but vegetatively dormant in 1989.

Missing1989 Marked as 1 if not found in 1989.

Seedling1989 Marked as 1, 2, or 3 if observed as a seedling in year *t*. Numbers refer to certainty of assignment: 1 = certain that plant is a seedling in 1989, 2 = likely that plant is a seedling in 1989, 3 = probable that plant is a seedling in 1989.

Volume90 Aboveground volume in mm³ in 1990.

lnVol90 Natural logarithm of `Volume90`.

FCODE90 Equals 1 if flowering and 0 if not flowering in 1990.

Flow90 Number of flowers in 1990.

Intactseed90 NZumber of intact mature seeds produced in 1990. Not always an integer, as in some cases seed number was estimated via linear modeling.

Dead1990 Marked as 1 if known to be dead in 1990.

Dormant1990 Marked as 1 if known to be alive but vegetatively dormant in 1990.

Missing1990 Marked as 1 if not found in 1990.

Seedling1990 Marked as 1, 2, or 3 if observed as a seedling in year *t*. Numbers refer to certainty of assignment: 1 = certain that plant is a seedling in 1990, 2 = likely that plant is a seedling in 1990, 3 = probable that plant is a seedling in 1990.

Volume91 Aboveground volume in mm³ in 1991.

lnVol91 Natural logarithm of `Volume91`.

FCODE91 Equals 1 if flowering and 0 if not flowering in 1991.

Flow91 Number of flowers in 1991.

Intactseed91 NZumber of intact mature seeds produced in 1991. Not always an integer, as in some cases seed number was estimated via linear modeling.

Dead1991 Marked as 1 if known to be dead in 1991.

Dormant1991 Marked as 1 if known to be alive but vegetatively dormant in 1991.

Missing1991 Marked as 1 if not found in 1991.

Seedling1991 Marked as 1, 2, or 3 if observed as a seedling in year *t*. Numbers refer to certainty of assignment: 1 = certain that plant is a seedling in 1991, 2 = likely that plant is a seedling in 1991, 3 = probable that plant is a seedling in 1991.

Source

Ehrlen, J. 2000. The dynamics of plant populations: does the history of individuals matter? *Ecology* 81(6):1675-1684.

Examples

```
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
matvector <- c(0, 0, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)

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

lathvert <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET",
  juvcol = c("Seedling1988", "Seedling1989", "Seedling1990", "Seedling1991"),
  sizeacol = c("Volume88", "Volume89", "Volume90", "Volume91"),
  repstracol = c("FCODE88", "FCODE89", "FCODE90", "FCODE91"),
  fecacol = c("Intactseed88", "Intactseed89", "Intactseed90", "Intactseed91"),
  deadacol = c("Dead1988", "Dead1989", "Dead1990", "Dead1991"),
  nonobsacol = c("Dormant1988", "Dormant1989", "Dormant1990", "Dormant1991"),
  censorcol = c("Missing1988", "Missing1989", "Missing1990", "Missing1991"),
  stageassign = lathframe, stagesize = "sizea",
  censorkeep = NA, censor = TRUE)

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "Sd", "Sdl", "mat"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "rep", "rep", "Sdl"),
  stage1 = c("Sd", "rep", "Sd", "rep", "npr", "npr", "Sd"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, "mat"),
  eststage2 = c(NA, NA, NA, NA, NA, NA, "Sdl"),
  eststage1 = c(NA, NA, NA, NA, NA, NA, "NotAlive"),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, 0.345, 0.054, NA),
  type = c(1, 1, 1, 1, 3, 3, 1), type_t12 = c(1, 2, 1, 2, 1, 1, 1),
  stageframe = lathframe, historical = TRUE)

ehrlen3 <- rlefko3(data = lathvert, stageframe = lathframe, year = "all",
  stages = c("stage3", "stage2", "stage1"), supplement = lathsupp3,
  yearcol = "year2", indivcol = "individ")

ehrlen3mean <- lmean(ehrlen3)
ehrlen3mean$A[[1]]
```

```
lambda3(ehrlen3mean)
```

`lmean` *Estimate Mean Projection Matrices*

Description

Function `lmean()` estimates mean projection matrices as element-wise arithmetic means. It produces `lefkoMat` objects if provided with them, or single matrices in a simple one-element list if provided a list of matrices. Will produce a `lefkoMatList` object, in which each element is a `lefkoMat` object, if provided with a `lefkoMatList` object.

Usage

```
lmean(mats, matsout = NULL, force_sparse = FALSE)
```

Arguments

`mats`	A `lefkoMat` object, a `lefkoMatList` object, or a list of square matrices of equal dimension in standard or sparse format.
`matsout`	A string identifying which means to estimate. Option `"pop"` indicates population-level only, `"patch"` indicates patch-level only, and `"all"` indicates that both patch- and population-level means should be estimated. Defaults to `"all"`.
`force_sparse`	A logical value identifying whether to output the mean matrices in sparse format, if input as standard matrices.

Value

Yields a `lefkoMat` object, a `lefkoMatList` object, or a list of matrices. If a `lefkoMat` object. then will have the following characteristics:

`A`	A list of full mean projection matrices in order of sorted populations, patches, and years. These are typically estimated as the sums of the associated mean `U` and `F` matrices. All matrices output in either the `matrix` class, or the `dgCMatrix` class.
`U`	A list of mean survival-transition matrices sorted as in `A`. All matrices output in the `matrix` class.
`F`	A list of mean fecundity matrices sorted as in `A`. All matrices output in the `matrix` class.
`hstages`	A data frame showing the pairing of ahistorical stages used to create historical stage pairs. Given if the MPM is historical.
`ahstages`	A data frame detailing the characteristics of associated ahistorical stages.
`labels`	A data frame detailing the order of population, patch, and year of each mean matrix. If `pop`, `patch`, or `year2` are `NA` in the original `labels` set, then these will be re-labeled as `A`, `1`, or `1`, respectively.

matrixqc	A short vector describing the number of non-zero elements in U and F mean matrices, and the number of annual matrices.
modelqc	This is the qc portion of the modelsuite input. Only output from lefkoMat objects resulting from function-based estimation.
dataqc	A vector showing the numbers of individuals and rows in the vertical dataset used as input. Only output from lefkoMat objects resulting from raw matrix estimation.

Examples

```
data(cypdata)

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 = 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", "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", patch = "all", stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added"), supplement = cypsupp2r,
  yearcol = "year2", patchcol = "patchid", indivcol = "individ")

cyp2mean <- lmean(cypmatrix2r)
```

`logistic3` *Two-Parameter logistic Function*

Description

Function `logistic3()` creates a vector of values produced by the logistic function as applied with a user-specified time lag. The logistic function is given as $\phi_{t+1} = \phi_t \lambda(1 - n_t/K)$. Here, if no `separate_N` vector is provided, then $n_t = \phi_t$. If λ is not provided, then it defaults to `1.0`.

Usage

```
logistic3(
  start_value,
  alpha,
  beta = 0,
  lambda = 1,
  time_steps = 100L,
  time_lag = 1L,
  pre0_subs = FALSE,
  pre0_value = 0,
  substoch = 0L,
  separate_N = NULL
)
```

Arguments

`start_value`	A positive number to start the return vector in time 0.
`alpha`	The carrying capacity K.
`beta`	If set to some positive number, then this number is the maximum value of phi to enforce. Otherwise, equals `0` and enforces no limit.
`lambda`	The value of the discrete population growth rate to use. Equal to the natural logarithm of the instantaneous growth rate, r.
`time_steps`	The number of time steps to run the projection. Must be a positive integer.
`time_lag`	A positive integer denoting the number of time steps back for the value of phi in the logistic function.
`pre0_subs`	A logical value indicating whether to use a number other than that given in `start_value` for values of phi lagged from times prior to time 0.
`pre0_value`	A positive number to use for phi lagged from times prior to time 0. Only used if `pre0_subs = TRUE`.
`substoch`	An integer value indicating the kind of substochasticity to use. Values include: `0`, no substochasticity enforced (the default); `1`, all numbers must be non-negative; and `2`, all numbers should be forced to the interval [0, 1].
`separate_N`	An optional numeric vector with values of N in each time, if phi is to be treated as different from N in the logistic model.

Value

A numeric vector of values showing values projected under the- logistic function.

Examples

```
trial_run1 <- logistic3(1, alpha = 5)
plot(trial_run1)

trial_run2 <- logistic3(1, alpha = 5, beta = 5)
plot(trial_run2)

trial_run3 <- logistic3(1, alpha = 100)
plot(trial_run3)

trial_run4 <- logistic3(1, alpha = 100, beta = 50)
plot(trial_run4)

trial_run5 <- logistic3(1, alpha = 500)
plot(trial_run5)

trial_run6 <- logistic3(1, alpha = 500, beta = 501)
plot(trial_run6)

used_Ns <- c(10, 15, 12, 14, 14, 150, 15, 1, 5, 7, 9, 14, 13, 16, 17, 19,
  25, 26)
trial_run7 <- logistic3(1, alpha = 500, beta = 501, separate_N = used_Ns)
plot(trial_run7)
```

---

ltre3 *Conduct a Life Table Response Experiment*

Description

`ltre3()` returns a set of matrices of one-way LTRE (life table response experiment), stochastic LTRE (sLTRE) matrices, or small noise approximation LTRE (sna-LTRE) contributions.

Usage

```
ltre3(
  mats,
  refmats = NA,
  ref = NA,
  stochastic = FALSE,
  times = 10000,
  burnin = 3000,
  tweights = NA,
  sparse = "auto",
  seed = NA,
```

```
  append_mats = FALSE,
  sna_ltre = FALSE,
  tol = 1e-30,
  ...
)
```

Arguments

mats	An object of class lefkoMat.
refmats	A reference lefkoMat object, or matrix, for use as the control. Default is NA, which sets to the same object as mats.
ref	A numeric value indicating which matrix or matrices in refmats to use as the control. The numbers used must correspond to the number of the matrices in the labels element of the associated lefkoMat object. The default setting, NA, uses all entries in refmats.
stochastic	A logical value determining whether to conduct a deterministic (FALSE) or stochastic (TRUE) elasticity analysis. Defaults to FALSE.
times	The number of occasions to project forward in stochastic simulation. Defaults to 10000.
burnin	The number of initial steps to ignore in stochastic projection when calculating stochastic elasticities. Must be smaller than steps. Defaults to 3000.
tweights	An optional numeric vector or matrix denoting the probabilities of choosing each matrix in a stochastic projection. If a matrix is input, 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 a vector is input, then the choice of annual matrix is assumed to be independent of the current matrix. Defaults to equal weighting among matrices. Note that SNA-LTRE analysis cannot take matrix input.
sparse	A text string indicating whether to use sparse matrix encoding ("yes") or dense matrix encoding ("no"). Defaults to "auto", in which case sparse matrix encoding is used with square matrices with at least 50 rows and no more than 50% of elements with values greater than zero.
seed	Optional numeric value corresponding to the random seed for stochastic simulation.
append_mats	A logical value denoting whether to include the original A, U, and F matrices in the returned lefkoLTRE object. Defaults to FALSE.
sna_ltre	A logical value indicating whether to treat stochastic LTRE via the sna-LTRE approach from Davison et al. (2019) (TRUE), or the stochastic LTRE approximation from Davison et al. (2010) (FALSE). Defaults to FALSE.
tol	A numeric value indicating a lower positive limit to matrix element values when applied to stochastic and small noise approximation LTRE estimation protocols. Matrix element values lower than this will be treated as 0.0 values. Defaults to 1e-30.
...	Other parameters.

Value

This function returns an object of class `lefkoLTRE`. This includes a list of LTRE matrices as object `cont_mean` if a deterministic LTRE is called for, or a list of mean-value LTRE matrices as object `cont_mean` and a list of SD-value LTRE matrices as object `cont_sd` if a stochastic LTRE is called for. If a small-noise approximation LTRE (SNA-LTRE) is performed, then the output includes six objects: `cont_mean`, which provides the contributions of shifts in mean matrix elements; `cont_elas`, which provides the contributions of shifts in the elasticities of matrix elements; `cont_cv`, which provides the contributions of temporal variation in matrix elements; `cont_corr`, which provides the contributions of temporal correlations in matrix elements; `r_values_m`, which provides a vector of log deterministic lambda values for treatment populations; and `r_values_ref`, which provides the log deterministic lambda of the mean reference matrix.This is followed by the stageframe as object `ahstages`, the order of historical stages as object `hstages`, the age-by-stage order as object `agestages`, the order of matrices as object `labels`, and, if requested, the original A, U, and F matrices.

Notes

Function `ltre3()` cannot handle `lefkoMatList` inputs, although individual `lefkoMat` elements within these lists can be used.

Deterministic LTRE is one-way, fixed, and based on the sensitivities of the matrix midway between each input matrix and the reference matrix, per Caswell (2001, Matrix Population Models, Sinauer Associates, MA, USA). Stochastic LTRE is performed via two methods. The stochastic LTRE approximation is simulated per Davison et al. (2010) Journal of Ecology 98:255-267 (doi: 10.1111/j.1365-2745.2009.01611.x). The small noise approximation (sna-LTRE) is analyzed per Davison et al. (2019) Ecological Modelling 408: 108760 (doi: 10.1016/j.ecolmodel.2019.108760).

All stochastic and small noise approximation LTREs conducted without reference matrices are performed as spatial tests of the population dynamics among patches.

Default behavior for stochastic LTRE uses the full population provided in `mats` as the reference if no `refmats` and `ref` is provided. If no `refmats` is provided but `ref` is, then the matrices noted in `ref` are used as the reference matrix set. Year and patch order is utilized from object `mats`, but not from object `refmats`, in which each matrix is assumed to represent a different year from one population. This function cannot currently handle multiple populations within the same `mats` object (although such analysis is possible if these populations are designated as patches instead).

If `sparse = "auto"`, the default, then sparse matrix encoding will be used if the size of the input matrices is at least 50 columns by 50 rows for deterministic and stochastic LTREs and 10 columns by 10 rows for small noise approximation LTREs, in all cases as long as 50% of the elements in the first matrix are non-zero.

Stochastic LTREs do not test for the impact of temporal change in vital rates. An MPM with a single population, a single patch, and only annual matrices will produce contributions of 0 to stochastic λ.

Speed can sometimes be increased by shifting from automatic sparse matrix determination to forced dense or sparse matrix projection. This will most likely occur when matrices have between 10 and 300 rows and columns. Defaults work best when matrices are very small and dense, or very large and sparse.

SNA-LTRE analysis cannot test the impact of first-order Markovian environments. However, different random weightings of annual matrices are allowed if given in vector format.

The time_weights, steps, force_sparse, and rseed arguments are now deprecated. Instead, please use the tweights, times, sparse, and seed arguments.

See Also

summary.lefkoLTRE()

Examples

```
data(cypdata)

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 = 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", "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", patch = "all", stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added"), supplement = cypsupp2r,
  yearcol = "year2", patchcol = "patchid", indivcol = "individ")

ltre3(cypmatrix2r, sna_ltre = TRUE)
```

markov_run *Creates Vector of Times Based on First-Order Markov Transition Matrix*

Description

Creates a vector of randomly sampled years / times to be used in projection. Random sampling requires a 1st order Markovian transition matrix, showing the probability of transitioning to each time from each time. Note that this function is not required if the probability of transitioning to a particular time does not vary with time.

Usage

```
markov_run(main_times, mat, times = 10000L, start = NULL)
```

Arguments

main_times	An integer vector giving the years / times to use.
mat	A matrix giving the transition probabilities from each time to each time. Must have the same number of columns and rows as there are elements in vector times.
times	The number of times to project forward. Defaults to 10000.
start	The start time to use. Defaults to the first time in vector main_times.

Value

An integer vector giving the order of times / years to use in projection. This can be used as input in the year option in functions projection3() and f_projection3().

See Also

projection3()

f_projection3()

Examples

```
# Cypripedium example
data(cypdata)

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 = 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)

cypsupp3r <- supplemental(stage3 = c("SD", "SD", "P1", "P1", "P2", "P3", "SL",
    "D", "XSm", "Sm", "D", "XSm", "Sm", "mat", "mat", "mat", "SD", "P1"),
  stage2 = c("SD", "SD", "SD", "SD", "P1", "P2", "P3", "SL", "SL", "SL", "SL",
    "SL", "SL", "D", "XSm", "Sm", "rep", "rep"),
  stage1 = c("SD", "rep", "SD", "rep", "SD", "P1", "P2", "P3", "P3", "P3",
    "SL", "SL", "SL", "SL", "SL", "SL", "mat", "mat"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, NA, "D", "XSm", "Sm", "D", "XSm", "Sm",
    "mat", "mat", "mat", NA, NA),
  eststage2 = c(NA, NA, NA, NA, NA, NA, NA, "XSm", "XSm", "XSm", "XSm", "XSm",
    "XSm", "D", "XSm", "Sm", NA, NA),
  eststage1 = c(NA, NA, NA, NA, NA, NA, NA, "XSm", "XSm", "XSm", "XSm", "XSm",
    "XSm", "XSm", "XSm", "XSm", NA, NA),
  givenrate = c(0.1, 0.1, 0.2, 0.2, 0.2, 0.2, 0.25, NA, NA, NA, NA, NA, NA,
    NA, NA, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
    NA, 0.5, 0.5),
  type = c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3),
  type_t12 = c(1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1),
  stageframe = cypframe_raw, historical = TRUE)

cypmatrix3r <- rlefko3(data = cypraw_v1, stageframe = cypframe_raw,
  year = "all", stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added", "size1added"),
  supplement = cypsupp3r, yearcol = "year2", indivcol = "individ")

used_years <-c(2005, 2006, 2007, 2008)

yr_tx_vec <- c(0.4, 0.2, 0.2, 0.2, 0.2, 0.4, 0.2, 0.2, 0.2, 0.2, 0.4, 0.2,
  0.2, 0.2, 0.2, 0.4)
yr_tx_mat <- matrix(yr_tx_vec, 4, 4)

set.seed(1)
cyp_markov_vec_1 <- markov_run(main_times = used_years, mat = yr_tx_mat,
  times = 100)

set.seed(2)
cyp_markov_vec_2 <- markov_run(main_times = used_years, mat = yr_tx_mat,
  times = 100)
```

```
set.seed(3)
cyp_markov_vec_3 <- markov_run(main_times = used_years, mat = yr_tx_mat,
  times = 100)

cypstoch_1 <- projection3(cypmatrix3r, nreps = 1, times = 100,
  year = cyp_markov_vec_1)
cypstoch_2 <- projection3(cypmatrix3r, nreps = 1, times = 100,
  year = cyp_markov_vec_2)
cypstoch_3 <- projection3(cypmatrix3r, nreps = 1, times = 100,
  year = cyp_markov_vec_3)
```

`matrix_interp` *Arranges Matrix Elements in Order of Magnitude for Interpretation*

Description

Function `matrix_interp` summarizes matrices from `lefkoMat`, `lefkoSens`, `lefkoElas`, and `lefkoLTRE` objects in terms of the magnitudes of their elements. It can also create ordered summaries of standard matrices and sparse matrices.

Usage

```
matrix_interp(object, mat_chosen = 1L, part = 1L, type = 3L)
```

Arguments

`object`	A list object in one of `lefko3`'s output formats, or a standard matrix or sparse matrix in `dgCMatrix` format. Standard `lefko3` output formats include `lefkoMat`, `lefkoSens`, `lefkoElas`, and `lefkoLTRE` objects.
`mat_chosen`	The number of the matrix to assess, within the appropriate matrix list. See `Notes` for further details.
`part`	An integer noting whether to provide assessments of which of the main types of matrices to analyze. In a standard `lefkoMat` object, the integers `1`, `2`, and `3` correspond to the `A`, `U`, and `F` lists, respectively. In `lefkoSens` and `lefkoElas` objects, the integers `1` and `2` correspond to the ahistorical matrix sets and the historical matrix sets, respectively. In deterministic and stochastic `lefkoLTRE` objects, the integers `1` and `2` correspond to the `cont_mean` and `cont_sd` lists, respectively.
`type`	An integer corresponding to the type of order summary, including most to least positive (`1`), most to least negative (`2`), and greatest to lowest absolute magnitude (`3`). Defaults to type `3`.

Value

A data frame arranging all elements in the matrix chosen from greatest and smallest. This can be a data frame of only positive elements, of only negative elements, or all elements in order of absolute magnitude.

Notes

Argument mat_chosen refers to the number of the matrix within the list that it is held in. For example, if the function is applied to the cont_sd portion of a stochastic LTRE, and there are four LTRE matrices within that list element corresponding to three patch LTRE matrices and one overall population-level LTRE matrix, then setting this value to 4 would focus the function on the overall population-level LTRE matrix associated with contributions of the standard deviations of elements. This argument should be left blank if a standard matrix or sparse matrix is input.

Huge sparse matrices may take more time to process than small, dense matrices.

Examples

```
data(cypdata)

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 = 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", "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", patch = "all", stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added"), supplement = cypsupp2r,
  yearcol = "year2", patchcol = "patchid", indivcol = "individ")
```

```
aaa <- ltre3(cypmatrix2r, stochastic = TRUE)

matrix_interp(aaa, mat_chosen = 1, part = 2, type = 3)
```

`miniMod` *Minimize lefkoMod Object by Conversion to vrm_input Object*

Description

This function takes a `lefkoMod` object, which consists of vital rate models, their associated `dredge` model tables, and related metadata, and converts them to minimal data frame lists useable in MPM creation and projection. The main advantage to using this approach is in memory savings.

Usage

```
miniMod(
  lMod,
  hfv_data = NULL,
  stageframe = NULL,
  all_years = NULL,
  all_patches = NULL,
  all_groups = NULL,
  all_indcova = NULL,
  all_indcovb = NULL,
  all_indcovc = NULL
)
```

Arguments

`lMod`	A `lefkoMod` object.
`hfv_data`	The `hfv_data` formatted data frame used to develop object `lMod`.
`stageframe`	The stageframe used to develop object `lMod`.
`all_years`	A vector giving the times / years used to develop object `lMod`, exactly as used in the latter. Only needed if object `hfv_data` not provided.
`all_patches`	A vector giving the patch names used to develop object `lMod`, exactly as used in the latter. Only needed if object `hfv_data` not provided.
`all_groups`	A vector giving the stage groups used to develop object `lMod`, exactly as used in the latter. Only needed if object `stageframe` not provided.
`all_indcova`	The name of individual covariate a if quantitative and non-categorical, or of the categories used if the covariate is a factor variable. Only needed if object `hfv_data` not provided but individual covariates used in vital rate models.
`all_indcovb`	The name of individual covariate a if quantitative and non-categorical, or of the categories used if the cvoariate is a factor variable. Only needed if object `hfv_data` not provided but individual covariates used in vital rate models.

all_indcovc The name of individual covariate a if quantitative and non-categorical, or of the categories used if the covariate is a factor variable. Only needed if object hfv_data not provided but individual covariates used in vital rate models.

Value

An object of class vrm_input. See function vrm_import() for details.

Examples

```
data(lathyrus)

sizevector <- c(0, 4.6, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 1, 2, 3, 4, 5, 6, 7, 8,
  9)
stagevector <- c("Sd", "Sdl", "Dorm", "Sz1nr", "Sz2nr", "Sz3nr", "Sz4nr",
  "Sz5nr", "Sz6nr", "Sz7nr", "Sz8nr", "Sz9nr", "Sz1r", "Sz2r", "Sz3r",
  "Sz4r", "Sz5r", "Sz6r", "Sz7r", "Sz8r", "Sz9r")
repvector <- c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1)
obsvector <- c(0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
matvector <- c(0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0)
indataset <- c(0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 4.6, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
  0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5)

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

lathvertln <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "lnVol88", repstracol = "Intactseed88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframeln, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, NAas0 = TRUE, censor = TRUE)

lathvertln$feca2 <- round(lathvertln$feca2)
lathvertln$feca1 <- round(lathvertln$feca1)
lathvertln$feca3 <- round(lathvertln$feca3)

lathmodelsln3 <- modelsearch(lathvertln, historical = TRUE,
  approach = "mixed", suite = "main",
  vitalrates = c("surv", "obs", "size", "repst", "fec"), juvestimate = "Sdl",
  bestfit = "AICc&k", sizedist = "gaussian", fecdist = "poisson",
  indiv = "individ", patch = "patchid", year = "year2",
  year.as.random = TRUE, patch.as.random = TRUE, show.model.tables = TRUE,
  quiet = "partial")

lathmodels_mini <- miniMod(lathmodelsln3, hfv_data = lathvertln,
```

```
  stageframe = lathframeln)
lathmodels_mini
```

`modelsearch` *Develop Best-fit Vital Rate Estimation Models for MPM Development*

Description

Function `modelsearch()` runs exhaustive model building and selection for each vital rate needed to estimate a function-based MPM or IPM. It returns best-fit models for each vital rate, model table showing all models tested, and model quality control data. The final output can be used as input in other functions within this package.

Usage

```
modelsearch(
  data,
  stageframe = NULL,
  historical = TRUE,
  approach = "mixed",
  suite = "size",
  interactions = FALSE,
  bestfit = "AICc&k",
  vitalrates = c("surv", "size", "fec"),
  surv = c("alive3", "alive2", "alive1"),
  obs = c("obsstatus3", "obsstatus2", "obsstatus1"),
  size = c("sizea3", "sizea2", "sizea1"),
  sizeb = c(NA, NA, NA),
  sizec = c(NA, NA, NA),
  repst = c("repstatus3", "repstatus2", "repstatus1"),
  fec = c("feca3", "feca2", "feca1"),
  stage = c("stage3", "stage2", "stage1"),
  matstat = c("matstatus3", "matstatus2", "matstatus1"),
  indiv = "individ",
  patch = NA,
  year = "year2",
  density = NA,
  test.density = FALSE,
  sizedist = "gaussian",
  sizebdist = NA,
  sizecdist = NA,
  fecdist = "gaussian",
  size.zero = FALSE,
  sizeb.zero = FALSE,
  sizec.zero = FALSE,
```

```
  size.trunc = FALSE,
  sizeb.trunc = FALSE,
  sizec.trunc = FALSE,
  fec.zero = FALSE,
  fec.trunc = FALSE,
  patch.as.random = TRUE,
  year.as.random = TRUE,
  juvestimate = NA,
  juvsize = FALSE,
  jsize.zero = FALSE,
  jsizeb.zero = FALSE,
  jsizec.zero = FALSE,
  jsize.trunc = FALSE,
  jsizeb.trunc = FALSE,
  jsizec.trunc = FALSE,
  fectime = 2,
  censor = NA,
  age = NA,
  test.age = FALSE,
  indcova = NA,
  indcovb = NA,
  indcovc = NA,
  random.indcova = FALSE,
  random.indcovb = FALSE,
  random.indcovc = FALSE,
  test.indcova = FALSE,
  test.indcovb = FALSE,
  test.indcovc = FALSE,
  annucova = NA,
  annucovb = NA,
  annucovc = NA,
  test.annucova = FALSE,
  test.annucovb = FALSE,
  test.annucovc = FALSE,
  test.group = FALSE,
  show.model.tables = TRUE,
  global.only = FALSE,
  accuracy = TRUE,
  data_out = FALSE,
  quiet = FALSE
)
```

Arguments

data	The vertical dataset to be used for analysis. This dataset should be of class `hfvdata`, but can also be a data frame formatted similarly to the output format provided by functions `verticalize3()` or `historicalize3()`, as long as all needed variables are properly designated. Can also be a list of class `hfvlist`, if bootstrapping is desired.

`stageframe` The stageframe characterizing the life history model used. Optional unless `test.group = TRUE`, in which case it is required. Defaults to `NULL`.

`historical` A logical variable denoting whether to assess the effects of state in occasion *t*-1, in addition to state in occasion *t*. Defaults to `TRUE`.

`approach` The statistical approach to be taken for model building. The default is `"mixed"`, which uses the mixed model approach utilized in packages `lme4` and `glmmTMB`. Other options include `"glm"`, which uses generalized linear modeling assuming that all factors are fixed.

`suite` Either a single string value or a vector of 14 strings for each vital rate model. Describes the global model for each vital rate estimation, and has the following possible values: `full`, includes main effects and all two-way interactions of size and reproductive status; `main`, includes main effects only of size and reproductive status; `size`, includes only size (also interactions between size in historical model); `rep`, includes only reproductive status (also interactions between status in historical model); `age`, all vital rates estimated with age and y-intercepts only; `cons`, all vital rates estimated only as y-intercepts. If `approach = "glm"` and `year.as.random = FALSE`, then year is also included as a fixed effect, and, if `interactions = TRUE`, is included in two-way interactions. Order of models in the string vector if more than 1 value is used is: 1) survival, 2) observation, 3) primary size, 4) secondary size, 5) tertiary size, 6) reproductive status, 7) fecundity, 8) juvenile survival, 9) juvenile observation, 10) juvenile primary size, 11) juvenile secondary size, 12) juvenile tertiary size, 13) juvenile reproductive status, and 14) juvenile maturity status. Defaults to `size`.

`interactions` A variable denoting whether to include two-way interactions between all fixed factors in the global model. Defaults to `FALSE`.

`bestfit` A variable indicating the model selection criterion for the choice of best-fit model. The default is `AICc&k`, which chooses the best-fit model as the model with the lowest AICc or, if not the same model, then the model that has the lowest degrees of freedom among models with $\Delta AICc <= 2.0$. Alternatively, `AICc` may be chosen, in which case the best-fit model is simply the model with the lowest AICc value.

`vitalrates` A vector describing which vital rates will be estimated via linear modeling, with the following options: `surv`, survival probability; `obs`, observation probability; `size`, overall size; `repst`, probability of reproducing; and `fec`, amount of reproduction (overall fecundity). May also be set to `vitalrates = "leslie"`, which is equivalent to setting `c("surv", "fec")` for a Leslie MPM. This choice also determines how internal data subsetting for vital rate model estimation will work. Defaults to `c("surv", "size", "fec")`.

`surv` A vector indicating the variable names coding for status as alive or dead in occasions *t*+1, *t*, and *t*-1, respectively. Defaults to `c("alive3", "alive2", "alive1")`.

`obs` A vector indicating the variable names coding for observation status in occasions *t*+1, *t*, and *t*-1, respectively. Defaults to `c("obsstatus3", "obsstatus2", "obsstatus1")`.

`size` A vector indicating the variable names coding for the primary size variable on occasions *t*+1, *t*, and *t*-1, respectively. Defaults to `c("sizea3", "sizea2", "sizea1")`.

`sizeb` A vector indicating the variable names coding for the secondary size variable on occasions *t*+1, *t*, and *t*-1, respectively. Defaults to `c(NA, NA, NA)`, in which case `sizeb` is not used.

`sizec` A vector indicating the variable names coding for the tertiary size variable on occasions *t*+1, *t*, and *t*-1, respectively. Defaults to `c(NA, NA, NA)`, in which case `sizec` is not used.

`repst` A vector indicating the variable names coding for reproductive status in occasions *t*+1, *t*, and *t*-1, respectively. Defaults to `c("repstatus3", "repstatus2", "repstatus1")`.

`fec` A vector indicating the variable names coding for fecundity in occasions *t*+1, *t*, and *t*-1, respectively. Defaults to `c("feca3", "feca2", "feca1")`.

`stage` A vector indicating the variable names coding for stage in occasions *t*+1, *t*, and *t*-1. Defaults to `c("stage3", "stage2", "stage1")`.

`matstat` A vector indicating the variable names coding for maturity status in occasions *t*+1, *t*, and *t*-1. Defaults to `c("matstatus3", "matstatus2", "matstatus1")`.

`indiv` A text value indicating the variable name coding individual identity. Defaults to `"individ"`.

`patch` A text value indicating the variable name coding for patch, where patches are defined as permanent subgroups within the study population. Defaults to `NA`.

`year` A text value indicating the variable coding for observation occasion *t*. Defaults to `year2`.

`density` A text value indicating the name of the variable coding for spatial density, should the user wish to test spatial density as a fixed factor affecting vital rates. Defaults to `NA`.

`test.density` Either a logical value indicating whether to include `density` as a fixed categorical variable in linear models, or a logical vector of such values for 14 models, in order: 1) survival, 2) observation, 3) primary size, 4) secondary size, 5) tertiary size, 6) reproductive status, 7) fecundity, 8) juvenile survival, 9) juvenile observation, 10) juvenile primary size, 11) juvenile secondary size, 12) juvenile tertiary size, 13) juvenile reproductive status, and 14) juvenile maturity status. Defaults to `FALSE`.

`sizedist` The probability distribution used to model primary size. Options include `"gaussian"` for the Normal distribution (default), `"poisson"` for the Poisson distribution, `"negbin"` for the negative binomial distribution (quadratic parameterization), and `"gamma"` for the Gamma distribution.

`sizebdist` The probability distribution used to model secondary size. Options include `"gaussian"` for the Normal distribution, `"poisson"` for the Poisson distribution, `"negbin"` for the negative binomial distribution (quadratic parameterization), and `"gamma"` for the Gamma distribution. Defaults to `NA`.

`sizecdist` The probability distribution used to model tertiary size. Options include `"gaussian"` for the Normal distribution, `"poisson"` for the Poisson distribution, `"negbin"` for the negative binomial distribution (quadratic parameterization), and `"gamma"` for the Gamma distribution. Defaults to `NA`.

fecdist — The probability distribution used to model fecundity. Options include "gaussian" for the Normal distribution (default), "poisson" for the Poisson distribution, "negbin" for the negative binomial distribution (quadratic parameterization), and "gamma" for the Gamma distribution.

size.zero — A logical variable indicating whether the primary size distribution should be zero-inflated. Only applies to Poisson and negative binomial distributions. Defaults to FALSE.

sizeb.zero — A logical variable indicating whether the secondary size distribution should be zero-inflated. Only applies to Poisson and negative binomial distributions. Defaults to FALSE.

sizec.zero — A logical variable indicating whether the tertiary size distribution should be zero-inflated. Only applies to Poisson and negative binomial distributions. Defaults to FALSE.

size.trunc — A logical variable indicating whether the primary size distribution should be zero-truncated. Only applies to Poisson and negative binomial distributions. Defaults to FALSE. Cannot be TRUE if size.zero = TRUE.

sizeb.trunc — A logical variable indicating whether the secondary size distribution should be zero-truncated. Only applies to Poisson and negative binomial distributions. Defaults to FALSE. Cannot be TRUE if sizeb.zero = TRUE.

sizec.trunc — A logical variable indicating whether the tertiary size distribution should be zero-truncated. Only applies to Poisson and negative binomial distributions. Defaults to FALSE. Cannot be TRUE if sizec.zero = TRUE.

fec.zero — A logical variable indicating whether the fecundity distribution should be zero-inflated. Only applies to Poisson and negative binomial distributions. Defaults to FALSE.

fec.trunc — A logical variable indicating whether the fecundity distribution should be zero-truncated. Only applies to the Poisson and negative binomial distributions. Defaults to FALSE. Cannot be TRUE if fec.zero = TRUE.

patch.as.random — If set to TRUE and approach = "mixed", then patch is included as a random factor. If set to FALSE and approach = "glm", then patch is included as a fixed factor. All other combinations of logical value and approach lead to patch not being included in modeling. Defaults to TRUE.

year.as.random — If set to TRUE and approach = "mixed", then year is included as a random factor. If set to FALSE, then year is included as a fixed factor. All other combinations of logical value and approach lead to year not being included in modeling. Defaults to TRUE.

juvestimate — An optional variable denoting the stage name of the juvenile stage in the vertical dataset. If not NA, and stage is also given (see below), then vital rates listed in vitalrates other than fec will also be estimated from the juvenile stage to all adult stages. Defaults to NA, in which case juvenile vital rates are not estimated.

juvsize — A logical variable denoting whether size should be used as a term in models involving transition from the juvenile stage. Defaults to FALSE, and is only used if juvestimate does not equal NA.

`jsize.zero` A logical variable indicating whether the primary size distribution of juveniles should be zero-inflated. Only applies to Poisson and negative binomial distributions. Defaults to `FALSE`.

`jsizeb.zero` A logical variable indicating whether the secondary size distribution of juveniles should be zero-inflated. Only applies to Poisson and negative binomial distributions. Defaults to `FALSE`.

`jsizec.zero` A logical variable indicating whether the tertiary size distribution of juveniles should be zero-inflated. Only applies to Poisson and negative binomial distributions. Defaults to `FALSE`.

`jsize.trunc` A logical variable indicating whether the primary size distribution in juveniles should be zero-truncated. Defaults to `FALSE`. Cannot be `TRUE` if `jsize.zero = TRUE`.

`jsizeb.trunc` A logical variable indicating whether the secondary size distribution in juveniles should be zero-truncated. Defaults to `FALSE`. Cannot be `TRUE` if `jsizeb.zero = TRUE`.

`jsizec.trunc` A logical variable indicating whether the tertiary size distribution in juveniles should be zero-truncated. Defaults to `FALSE`. Cannot be `TRUE` if `jsizec.zero = TRUE`.

`fectime` A variable indicating which year of fecundity to use as the response term in fecundity models. Options include `2`, which refers to occasion *t*, and `3`, which refers to occasion *t*+1. Defaults to `2`.

`censor` A vector denoting the names of censoring variables in the dataset, in order from occasion *t*+1, followed by occasion *t*, and lastly followed by occasion *t*-1. Defaults to `NA`.

`age` Designates the name of the variable corresponding to age in time *t* in the vertical dataset. Defaults to `NA`, in which case age is not included in linear models. Should only be used if building Leslie or age x stage matrices.

`test.age` Either a logical value indicating whether to include `age` as a fixed categorical variable in linear models, or a logical vector of such values for 14 models, in order: 1) survival, 2) observation, 3) primary size, 4) secondary size, 5) tertiary size, 6) reproductive status, 7) fecundity, 8) juvenile survival, 9) juvenile observation, 10) juvenile primary size, 11) juvenile secondary size, 12) juvenile tertiary size, 13) juvenile reproductive status, and 14) juvenile maturity status. Defaults to `FALSE`.

`indcova` Vector designating the names in occasions *t*+1, *t*, and *t*-1 of an individual covariate. Defaults to `NA`.

`indcovb` Vector designating the names in occasions *t*+1, *t*, and *t*-1 of a second individual covariate. Defaults to `NA`.

`indcovc` Vector designating the names in occasions *t*+1, *t*, and *t*-1 of a third individual covariate. Defaults to `NA`.

`random.indcova` A logical value indicating whether `indcova` should be treated as a random categorical factor, rather than as a fixed factor. Defaults to `FALSE`.

`random.indcovb` A logical value indicating whether `indcovb` should be treated as a random categorical factor, rather than as a fixed factor. Defaults to `FALSE`.

random.indcovc A logical value indicating whether indcovc should be treated as a random categorical factor, rather than as a fixed factor. Defaults to FALSE.

test.indcova Either a logical value indicating whether to include the indcova variable as a fixed categorical variable in linear models, or a logical vector of such values for 14 models, in order: 1) survival, 2) observation, 3) primary size, 4) secondary size, 5) tertiary size, 6) reproductive status, 7) fecundity, 8) juvenile survival, 9) juvenile observation, 10) juvenile primary size, 11) juvenile secondary size, 12) juvenile tertiary size, 13) juvenile reproductive status, and 14) juvenile maturity status. Defaults to FALSE.

test.indcovb Either a logical value indicating whether to include the indcovb variable as a fixed categorical variable in linear models, or a logical vector of such values for 14 models, in order: 1) survival, 2) observation, 3) primary size, 4) secondary size, 5) tertiary size, 6) reproductive status, 7) fecundity, 8) juvenile survival, 9) juvenile observation, 10) juvenile primary size, 11) juvenile secondary size, 12) juvenile tertiary size, 13) juvenile reproductive status, and 14) juvenile maturity status. Defaults to FALSE.

test.indcovc Either a logical value indicating whether to include the indcovc variable as a fixed categorical variable in linear models, or a logical vector of such values for 14 models, in order: 1) survival, 2) observation, 3) primary size, 4) secondary size, 5) tertiary size, 6) reproductive status, 7) fecundity, 8) juvenile survival, 9) juvenile observation, 10) juvenile primary size, 11) juvenile secondary size, 12) juvenile tertiary size, 13) juvenile reproductive status, and 14) juvenile maturity status. Defaults to FALSE.

annucova A numeric vector of annual covariates to test within all vital rate models. If historical = TRUE, then the number of elements must be equal to the number of values of year2 in the dataset plus one, and the vector should start with the value to be used in time *t*-1 in the first year of the dataset. In all other cases, the length of the vector must equal the number of values of year2 in the dataset. Defaults to NA.

annucovb A second numeric vector of annual covariates to test within all vital rate models. If historical = TRUE, then the number of elements must be equal to the number of values of year2 in the dataset plus one, and the vector should start with the value to be used in time *t*-1 in the first year of the dataset. In all other cases, the length of the vector must equal the number of values of year2 in the dataset. Defaults to NA.

annucovc A third numeric vector of annual covariates to test within all vital rate models. If historical = TRUE, then the number of elements must be equal to the number of values of year2 in the dataset plus one, and the vector should start with the value to be used in time *t*-1 in the first year of the dataset. In all other cases, the length of the vector must equal the number of values of year2 in the dataset. Defaults to NA.

test.annucova A logical value indicating whether to test the variable given as annucova as a fixed factor in analysis. Defaults to FALSE.

test.annucovb A logical value indicating whether to test the variable given as annucovb as a fixed factor in analysis. Defaults to FALSE.

test.annucovc A logical value indicating whether to test the variable given as annucovc as a fixed factor in analysis. Defaults to FALSE.

`test.group` Either a logical value indicating whether to include the `group` variable from the input `stageframe` as a fixed categorical variable in linear models, or a logical vector of such values for 14 models, in order: 1) survival, 2) observation, 3) primary size, 4) secondary size, 5) tertiary size, 6) reproductive status, 7) fecundity, 8) juvenile survival, 9) juvenile observation, 10) juvenile primary size, 11) juvenile secondary size, 12) juvenile tertiary size, 13) juvenile reproductive status, and 14) juvenile maturity status. Defaults to `FALSE`.

`show.model.tables` If set to TRUE, then includes full modeling tables in the output. Defaults to `TRUE`.

`global.only` If set to TRUE, then only global models will be built and evaluated. Defaults to `FALSE`.

`accuracy` A logical value indicating whether to test accuracy of models. See `Notes` section for details on how accuracy is assessed. Defaults to `TRUE`.

`data_out` A logical value indicating whether to append all subsetted datasets used in model building and selection to the output. Defaults to `FALSE`.

`quiet` May be a logical value, or any one of the strings `"yes"`, `"no"`, or `"partial"`. If set to `TRUE` or `"yes"`, then model building and selection will proceed with most warnings and diagnostic messages silenced. If set to `FALSE` or `"no"`, then all warnings and diagnostic messages will be displayed. If set to `"partial"`, then only messages related to transitions between different vital rate models will be displayed. Defaults to `FALSE`.

Value

This function yields an object of class `lefkoMod`, or a list of class `lefkoModList` if the data entered is of class `hfvlist`. If the latter, then each element is an object of class `lefkoMod`. Class `lefkoMod` objects are themselves lists. The first 14 elements are the best-fit models for survival, observation status, primary size, secondary size, tertiary size, reproductive status, fecundity, juvenile survival, juvenile observation, juvenile primary size, juvenile secondary size, juvenile tertiary size, juvenile transition to reproduction, and juvenile transition to maturity, respectively. This is followed by 14 elements corresponding to the model tables for each of these vital rates, in order, followed by a data frame showing the order and names of variables used in modeling, followed by a single character element denoting the criterion used for model selection, and ending on a data frame with quality control data:

`survival_model` Best-fit model of the binomial probability of survival from occasion *t* to occasion *t*+1. Defaults to 1.

`observation_model` Best-fit model of the binomial probability of observation in occasion *t*+1 given survival to that occasion. Defaults to 1.

`size_model` Best-fit model of the primary size metric on occasion *t*+1 given survival to and observation in that occasion. Defaults to 1.

`sizeb_model` Best-fit model of the secondary size metric on occasion *t*+1 given survival to and observation in that occasion. Defaults to 1.

`sizec_model` Best-fit model of the tertiary size metric on occasion *t*+1 given survival to and observation in that occasion. Defaults to 1.

`repstatus_model`
: Best-fit model of the binomial probability of reproduction in occasion t+1, given survival to and observation in that occasion. Defaults to 1.

`fecundity_model`
: Best-fit model of fecundity in occasion t+1 given survival to, and observation and reproduction in that occasion. Defaults to 1.

`juv_survival_model`
: Best-fit model of the binomial probability of survival from occasion t to occasion t+1 of an immature individual. Defaults to 1.

`juv_observation_model`
: Best-fit model of the binomial probability of observation in occasion t+1 given survival to that occasion of an immature individual. Defaults to 1.

`juv_size_model`
: Best-fit model of the primary size metric on occasion t+1 given survival to and observation in that occasion of an immature individual. Defaults to 1.

`juv_sizeb_model`
: Best-fit model of the secondary size metric on occasion t+1 given survival to and observation in that occasion of an immature individual. Defaults to 1.

`juv_sizec_model`
: Best-fit model of the tertiary size metric on occasion t+1 given survival to and observation in that occasion of an immature individual. Defaults to 1.

`juv_reproduction_model`
: Best-fit model of the binomial probability of reproduction in occasion t+1, given survival to and observation in that occasion of an individual that was immature in occasion t. This model is technically not a model of reproduction probability for individuals that are immature, rather reproduction probability here is given for individuals that are mature in occasion t+1 but immature in occasion t. Defaults to 1.

`juv_maturity_model`
: Best-fit model of the binomial probability of becoming mature in occasion t+1, given survival to that occasion of an individual that was immature in occasion t. Defaults to 1.

`survival_table`
: Full dredge model table of survival probability.

`observation_table`
: Full dredge model table of observation probability.

`size_table`
: Full dredge model table of the primary size variable.

`sizeb_table`
: Full dredge model table of the secondary size variable.

`sizec_table`
: Full dredge model table of the tertiary size variable.

`repstatus_table`
: Full dredge model table of reproduction probability.

`fecundity_table`
: Full dredge model table of fecundity.

`juv_survival_table`
: Full dredge model table of immature survival probability.

`juv_observation_table`
: Full dredge model table of immature observation probability.

`juv_size_table`
: Full dredge model table of primary size in immature individuals.

`juv_sizeb_table`
: Full dredge model table of secondary size in immature individuals.

`juv_sizec_table`
: Full dredge model table of tertiary size in immature individuals.

`juv_reproduction_table`
: Full dredge model table of immature reproduction probability.

`juv_maturity_table`
: Full dredge model table of the probability of an immature individual transitioning to maturity.

`paramnames`
: A data frame showing the names of variables from the input data frame used in modeling, their associated standardized names in linear models, and a brief comment describing each variable.

`criterion`
: Character variable denoting the criterion used to determine the best-fit model.

`qc`
: Data frame with five variables: 1) Name of vital rate, 2) number of individuals used to model that vital rate, 3) number of individual transitions used to model that vital rate, 4) parameter distribution used to model the vital rats, and 5) accuracy of model, given as detailed in Notes section.

`subdata`
: An optional list of data frames, each of which is the data frame used to develop each model in the `lefkoMod` object, in order.

Notes

When `modelsearch()` is called, it first trims the dataset down to just the variables that will be used (including all response terms and independent variables). It then subsets the data to only complete cases for those variables. Next, it builds global models for all vital rates, and runs them. If a global model fails, then the function proceeds by dropping terms. If `approach = "mixed"`, then it will determine which random factor term contains the most categories in the respective subset, and drop that term. If this fails, or if `approach = "glm"`, then it will drop any two-way interactions and run the model. If this fails, then the function will attempt to drop further terms, first patch alone, then year alone, then individual covariates by themselves, then combinations of these four, and finally individual identity. If all of these attempts fail and the approach used is `mixed`, then the function will try running a glm version of the original failed model. Finally, if all attempts fail, then the function displays a warning and returns `1` to allow model building assuming a constant rate or probability.

Including annual covariates is easy via the arguments `annucova`, `annucovb`, and `annucovc` together with `test.annucova`, `test.annucovb`, and `test.annucovc`. Rather than incorporate a nnual covariates into the dataset, the values corresponding to each year may be concatenated into a numeric vector, and then used in one of these three arguments. Function `modelsearch()` will then append the value associated with each year into the dataset, and proceed with model building. Two-way interactions can be explored with other main effects fixed variables by setting `interactions = TRUE`.

Setting `suite = "cons"` prevents the inclusion of size and reproductive status as fixed, independent factors in modeling. However, it does not prevent any other terms from being included. Density, age, individual covariates, individual identity, patch, and year may all be included.

The mechanics governing model building are fairly robust to errors and exceptions. The function attempts to build global models, and simplifies models automatically should model building fail. Model building proceeds through the functions `lm()` (GLM with Gaussian response), `glm()` (GLM

with Poisson, Gamma, or binomial response), glm.nb() (GLM with negative binomial response), zeroinfl() (GLM with zero-inflated Poisson or negative binomial response), vglm() (GLM with zero-truncated Poisson or negative binomial response), lmer() (mixed model with Gaussian response), glmer() (mixed model with binomial, Poisson, or Gamma response), and glmmTMB() (mixed model with negative binomial, or zero-truncated or zero-inflated Poisson or negative binomial response). See documentation related to these functions for further information. Any response term that is invariable in the dataset will lead to a best-fit model for that response represented by a single constant value.

Exhaustive model building and selection proceeds via the dredge() function in package MuMIn. This function is verbose, so that any errors and warnings developed during model building, model analysis, and model selection can be found and dealt with. Interpretations of errors during global model analysis may be found in documentation for the functions and packages mentioned. Package MuMIn is used for model dredging (see dredge()), and errors and warnings during dredging can be interpreted using the documentation for that package. Errors occurring during dredging lead to the adoption of the global model as the best-fit, and the user should view all logged errors and warnings to determine the best way to proceed. The quiet = TRUE and quiet = "partial" options can be used to silence dredge warnings, but users should note that automated model selection can be viewed as a black box, and so care should be taken to ensure that the models run make biological sense, and that model quality is prioritized.

Exhaustive model selection through dredging works best with larger datasets and fewer tested parameters. Setting suite = "full" and interactions = TRUE may initiate a dredge that takes a dramatically long time, particularly if the model is historical, individual or annual covariates are used, or a zero-inflated distribution is assumed. In such cases, the number of models built and tested will run at least in the millions. Small datasets will also increase the error associated with these tests, leading to adoption of simpler models overall. Note also that zero-inflated models are processed as two models, and so include twice the assumed number of parameters. If suite = "full", then this function will switch to a main effects global model for the zero-inflated parameter models if the total number of parameters to test rises above the limits imposed by the dredge() function in package MuMIn.

Accuracy of vital rate models is calculated differently depending on vital rate and assumed distribution. For all vital rates assuming a binomial distribution, including survival, observation status, reproductive status, and juvenile version of these, accuracy is calculated as the percent of predicted responses equal to actual responses. In all other models, accuracy is actually assessed as a simple R-squared in which the observed response values per data subset are compared to the predicted response values according to each best-fit model. Note that some situations in which factor variables are used may result in failure to assess accuracy. In these cases, function modelsearch() simply yields NA values.

Care must be taken to build models that test the impacts of state in occasion t-1 for historical models, and that do not test these impacts for ahistorical models. Ahistorical matrix modeling particularly will yield biased transition estimates if historical terms from models are ignored. This can be dealt with at the start of modeling by setting historical = FALSE for the ahistorical case, and historical = TRUE for the historical case.

This function handles generalized linear models (GLMs) under zero-inflated distributions using the zeroinfl() function, and zero- truncated distributions using the vglm() function. Model dredging may fail with these functions, leading to the global model being accepted as the best-fit model. However, model dredges of mixed models work for all distributions. We encourage the use of mixed models in all cases.

The negative binomial and truncated negative binomial distributions use the quadratic structure emphasized in Hardin and Hilbe (2018, 4th Edition of Generalized Linear Models and Extensions). The truncated negative binomial distribution may fail to predict size probabilities correctly when dispersion is near that expected of the Poisson distribution. To prevent this problem, we have integrated a cap on the overdispersion parameter. However, when using this distribution, please check the matrix column sums to make sure that they do not predict survival greater than 1.0. If they do, then please use either the negative binomial distribution or the zero-truncated Poisson distribution.

If density dependence is explored through function `modelsearch()`, then the interpretation of density is not the full population size but rather the spatial density term included in the dataset.

Users building vital rate models for Leslie matrices must set `vitalrates = c("surv", "fec")` or `vitalrates = "leslie"` rather than the default, because only survival and fecundity should be estimated in these cases. Also, the `suite` setting can be set to either `age` or `cons`, as the results will be exactly the same.

Users wishing to test age, density, group, or individual covariates, must include `test.age = TRUE`, `test.density = TRUE`, `test.group = TRUE`, or `test.indcova = TRUE` (or `test.indcovb = TRUE` or `test.indcovc = TRUE`, whichever is most appropriate), respectively, in addition to stipulating the name of the variable within the dataset. The default for these options is always `FALSE`.

`lefkoMod` objects can be quite large when datasets are large and models are complicated. To reduce the amount of memory taken up by models, `vrm_input` objects can be created to summarize all relevant aspects of the vital rate models using function `miniMod()`.

Examples

```
data(lathyrus)

sizevector <- c(0, 4.6, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 1, 2, 3, 4, 5, 6, 7, 8,
  9)
stagevector <- c("Sd", "Sdl", "Dorm", "Sz1nr", "Sz2nr", "Sz3nr", "Sz4nr",
  "Sz5nr", "Sz6nr", "Sz7nr", "Sz8nr", "Sz9nr", "Sz1r", "Sz2r", "Sz3r",
  "Sz4r", "Sz5r", "Sz6r", "Sz7r", "Sz8r", "Sz9r")
repvector <- c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1)
obsvector <- c(0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
matvector <- c(0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0)
indataset <- c(0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 4.6, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
  0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5)

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

lathvertln <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "lnVol88", repstracol = "Intactseed88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
```

```
  nonobsacol = "Dormant1988", stageassign = lathframeln, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, NAas0 = TRUE, censor = TRUE)

lathvertln$feca2 <- round(lathvertln$feca2)
lathvertln$feca1 <- round(lathvertln$feca1)
lathvertln$feca3 <- round(lathvertln$feca3)

lathmodelsln3 <- modelsearch(lathvertln, historical = TRUE,
  approach = "mixed", suite = "main",
  vitalrates = c("surv", "obs", "size", "repst", "fec"), juvestimate = "Sdl",
  bestfit = "AICc&k", sizedist = "gaussian", fecdist = "poisson",
  indiv = "individ", patch = "patchid", year = "year2", year.as.random = TRUE,
  patch.as.random = TRUE, show.model.tables = TRUE, quiet = "partial")

# Here we use supplemental() to provide overwrite and reproductive info
lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "mat", "Sd", "Sdl"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "Sdl", "rep", "rep"),
  stage1 = c("Sd", "rep", "Sd", "rep", "Sd", "mat", "mat"),
  eststage3 = c(NA, NA, NA, NA, "mat", NA, NA),
  eststage2 = c(NA, NA, NA, NA, "Sdl", NA, NA),
  eststage1 = c(NA, NA, NA, NA, "Sdl", NA, NA),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, NA, 0.345, 0.054),
  type = c(1, 1, 1, 1, 1, 3, 3), type_t12 = c(1, 2, 1, 2, 1, 1, 1),
  stageframe = lathframeln, historical = TRUE)

lathmat3ln <- flefko3(year = "all", patch = "all", stageframe = lathframeln,
  modelsuite = lathmodelsln3, data = lathvertln, supplement = lathsupp3,
  reduce = FALSE)
```

`mpm_create` *General Matrix Projection Model and Bootstrapped MPM Creation*

Description

Function `mpm_create()` is the core workhorse function that creates all flavors of MPM in `lefko3`. All other MPM creation functions act as wrappers for this function.

Usage

```
mpm_create(
  historical = FALSE,
  stage = TRUE,
  age = FALSE,
  devries = FALSE,
  reduce = FALSE,
  simple = FALSE,
```

```
  err_check = FALSE,
  data = NULL,
  year = NULL,
  pop = NULL,
  patch = NULL,
  stageframe = NULL,
  supplement = NULL,
  overwrite = NULL,
  repmatrix = NULL,
  alive = NULL,
  obsst = NULL,
  size = NULL,
  sizeb = NULL,
  sizec = NULL,
  repst = NULL,
  matst = NULL,
  fec = NULL,
  stages = NULL,
  yearcol = NULL,
  popcol = NULL,
  patchcol = NULL,
  indivcol = NULL,
  agecol = NULL,
  censorcol = NULL,
  modelsuite = NULL,
  paramnames = NULL,
  inda = NULL,
  indb = NULL,
  indc = NULL,
  annua = NULL,
  annub = NULL,
  annuc = NULL,
  dev_terms = NULL,
  density = NA_real_,
  CDF = TRUE,
  random_inda = FALSE,
  random_indb = FALSE,
  random_indc = FALSE,
  negfec = FALSE,
  exp_tol = 700L,
  theta_tol = 100000000L,
  censor = FALSE,
  censorkeep = NULL,
  start_age = NA_integer_,
  last_age = NA_integer_,
  fecage_min = NA_integer_,
  fecage_max = NA_integer_,
  fectime = 2L,
```

```
  fecmod = 1,
  cont = TRUE,
  prebreeding = TRUE,
  stage_NRasRep = FALSE,
  initial_nan = FALSE,
  sparse_output = FALSE
)
```

Arguments

historical	A logical value indicating whether to build a historical MPM. Defaults to FALSE.
stage	A logical value indicating whether to build a stage-based MPM. If both stage = TRUE and age = TRUE, then will proceed to build an age-by-stage MPM. Defaults to TRUE.
age	A logical value indicating whether to build an age-based MPM. If both stage = TRUE and age = TRUE, then will proceed to build an age-by-stage MPM. Defaults to FALSE.
devries	A logical value indicating whether to use deVries format for historical MPMs. Defaults to FALSE, in which case historical MPMs are created in Ehrlen format.
reduce	A logical value denoting whether to remove ages, ahistorical stages, or historical stages associated exclusively with zero transitions. These are removed only if the respective row and column sums in ALL matrices estimated equal 0. Defaults to FALSE.
simple	A logical value indicating whether to produce A, U, and F matrices, or only the latter two. Defaults to FALSE, in which case all three are output.
err_check	A logical value indicating whether to append extra information used in matrix calculation within the output list. Defaults to FALSE.
data	A data frame of class hfvdata. Required for all MPMs, except for function-based MPMs in which modelsuite is set to a vrm_input object.
year	A variable corresponding to observation occasion, or a set of such values, given in values associated with the year term used in vital rate model development. Can also equal "all", in which case matrices will be estimated for all occasions. Defaults to "all".
pop	A variable designating which populations will have matrices estimated. Should be set to specific population names, or to "all" if all populations should have matrices estimated. Only used in raw MPMs.
patch	A variable designating which patches or subpopulations will have matrices estimated. Should be set to specific patch names, or to "all" if matrices should be estimated for all patches. Defaults to NULL, in which case patch designations are ignored.
stageframe	An object of class stageframe. These objects are generated by function sf_create(), and include information on the size, observation status, propagule status, reproduction status, immaturity status, maturity status, stage group, size bin widths, and other key characteristics of each ahistorical stage. Not needed for purely age-based MPMs.

`supplement` An optional data frame of class `lefkoSD` that provides supplemental data that should be incorporated into the MPM. Three kinds of data may be integrated this way: transitions to be estimated via the use of proxy transitions, transition overwrites from the literature or supplemental studies, and transition multipliers for survival and fecundity. This data frame should be produced using the `supplemental()` function. Can be used in place of or in addition to an overwrite table (see `overwrite` below) and a reproduction matrix (see `repmatrix` below).

`overwrite` An optional data frame developed with the `overwrite()` function describing transitions to be overwritten either with given values or with other estimated transitions. Note that this function supplements overwrite data provided in `supplement`.

`repmatrix` An optional reproduction matrix. This matrix is composed mostly of `0`s, with non-zero entries acting as element identifiers and multipliers for fecundity (with `1` equaling full fecundity). If left blank, and no `supplement` is provided, then all stages marked as reproductive produce offspring at 1x that of estimated fecundity, and that offspring production will yield the first stage noted as propagule or immature. May be the dimensions of either a historical or an ahistorical matrix. If the latter, then all stages will be used in occasion *t*-1 for each suggested ahistorical transition. Not used in purely age-based MPMs.

`alive` A vector of names of binomial variables corresponding to status as alive (`1`) or dead (`0`) in occasions *t*+1, *t*, and *t*-1, respectively. Defaults to `c("alive3", "alive2", "alive1")` for historical MPMs, and `c("alive3", "alive2")` for ahistorical MPMs. Only needed for raw MPMs.

`obsst` A vector of names of binomial variables corresponding to observation status in occasions *t*+1, *t*, and *t*-1, respectively. Defaults to `c("obsstatus3", "obsstatus2", "obsstatus1")` for historical MPMs, and `c("obsstatus3", "obsstatus2")` for ahistorical MPMs. Only needed for raw MPMs.

`size` A vector of names of variables coding the primary size variable in occasions *t*+1, *t*, and *t*-1, respectively. Defaults to `c("sizea3", "sizea2", "sizea1")` for historical MPMs, and `c("sizea3", "sizea2")` for ahistorical MPMs. Only needed for raw, stage-based MPMs.

`sizeb` A vector of names of variables coding the secondary size variable in occasions *t*+1, *t*, and *t*-1, respectively. Defaults to an empty set, assuming that secondary size is not used. Only needed for raw, stage-based MPMs.

`sizec` A vector of names of variables coding the tertiary size variable in occasions *t*+1, *t*, and *t*-1, respectively. Defaults to an empty set, assuming that tertiary size is not used. Only needed for raw, stage-based MPMs.

`repst` A vector of names of binomial variables corresponding to reproductive status in occasions *t*+1, *t*, and *t*-1, respectively. Defaults to `c("repstatus3", "repstatus2", "repstatus1")` for historical MPMs, and `c("repstatus3", "repstatus2")` for ahistorical MPMs. Only needed for raw MPMs.

`matst` A vector of names of binomial variables corresponding to maturity status in occasions *t*+1, *t*, and *t*-1, respectively. Defaults to `c("matstatus3", "matstatus2", "matstatus1")` for historical MPMs, and `c("matstatus3", "matstatus2")` for ahistorical MPMs. Must be provided if building raw MPMs, and `stages` is not provided.

`fec` A vector of names of variables coding for fecundity in occasions *t*+1, *t*, and *t*-1, respectively. Defaults to `c("feca3", "feca2", "feca1")` for historical MPMs, and `c("feca3", "feca2")` for ahistorical MPMs. Only needed for raw, stage-based MPMs.

`stages` An optional vector denoting the names of the variables within the main vertical dataset coding for the stages of each individual in occasions *t*+1 and *t*, and *t*-1, if historical. The names of stages in these variables should match those used in the `stageframe` exactly. If left blank, then `rlefko3()` will attempt to infer stages by matching values of `alive`, `obsst`, `size`, `sizev`, `sizec`, `repst`, and `matst` to characteristics noted in the associated `stageframe`. Only used in raw, stage-based MPMs.

`yearcol` The variable name or column number corresponding to occasion *t* in the dataset. Defaults to `"year2"`. Only needed for raw MPMs.

`popcol` The variable name or column number corresponding to the identity of the population. Defaults to `"popid"` if a value is provided for `pop`; otherwise empty. Only needed for raw MPMs.

`patchcol` The variable name or column number corresponding to patch in the dataset. Defaults to `"patchid"` if a value is provided for `patch`; otherwise empty. Only needed for raw MPMs.

`indivcol` The variable name or column number coding individual identity. Only needed for raw MPMs.

`agecol` The variable name or column corresponding to age in time *t*. Defaults to `"obsage"`. Only used in raw age-based and age-by-stage MPMs.

`censorcol` The variable name or column number denoting the censor status. Only needed in raw MPMs, and only if `censor = TRUE`.

`modelsuite` One of three kinds of lists. The first is a `lefkoMod` object holding the vital rate models and associated metadata. Alternatively, an object of class `vrm_input` may be provided. Finally, this argument may simply be a list of models used to parameterize the MPM. In the final scenario, `data` and `paramnames` must also be given, and all variable names must match across all objects. If entered, then a function-based MPM will be developed. Otherwise, a raw MPM will be developed. Only used in function-based MPMs.

`paramnames` A data frame with three columns, the first describing all terms used in linear modeling, the second (must be called `mainparams`) giving the general model terms that will be used in matrix creation, and the third showing the equivalent terms used in modeling (must be named `modelparams`). Function `create_pm()` can be used to create a skeleton `paramnames` object, which can then be edited. Only required to build function-based MPMs if `modelsuite` is neither a `lefkoMod` object nor a `vrm_input` object.

`inda` Can be a single value to use for individual covariate `a` in all matrices, a pair of values to use for times *t* and *t*-1 in historical matrices, or a vector of such values corresponding to each occasion in the dataset. Defaults to `NULL`. Only used in function-based MPMs.

`indb` Can be a single value to use for individual covariate `b` in all matrices, a pair of values to use for times *t* and *t*-1 in historical matrices, or a vector of such values

corresponding to each occasion in the dataset. Defaults to NULL. Only used in function-based MPMs.

indc — Can be a single value to use for individual covariate c in all matrices, a pair of values to use for times *t* and *t*-1 in historical matrices, or a vector of such values corresponding to each occasion in the dataset. Defaults to NULL. Only used in function-based MPMs.

annua — Can be a single value to use for annual covariate a in all matrices, a pair of values to use for times *t* and *t*-1 in historical matrices, or a vector of such values corresponding to each occasion in the dataset. Defaults to NULL. Only used in function-based MPMs.

annub — Can be a single value to use for annual covariate b in all matrices, a pair of values to use for times *t* and *t*-1 in historical matrices, or a vector of such values corresponding to each occasion in the dataset. Defaults to NULL. Only used in function-based MPMs.

annuc — Can be a single value to use for annual covariate c in all matrices, a pair of values to use for times *t* and *t*-1 in historical matrices, or a vector of such values corresponding to each occasion in the dataset. Defaults to NULL. Only used in function-based MPMs.

dev_terms — A numeric vector of 2 elements in the case of a Leslie MPM, and of 14 elements in all other cases. Consists of scalar additions to the y-intercepts of vital rate linear models used to estimate vital rates in function-based MPMs. Defaults to 0 values for all vital rates.

density — A numeric value indicating density value to use to propagate matrices. Only needed if density is an explanatory term used in one or more vital rate models. Defaults to NA. Only used in function_based MPMs.

CDF — A logical value indicating whether to use the cumulative distribution function to estimate size transition probabilities in function-based MPMs. Defaults to TRUE, and should only be changed to FALSE if approximate probabilities calculated via the midpoint method are preferred.

random_inda — A logical value denoting whether to treat individual covariate a as a random, categorical variable. Otherwise is treated as a fixed, numeric variable. Defaults to FALSE. Only used in function-based MPMs.

random_indb — A logical value denoting whether to treat individual covariate b as a random, categorical variable. Otherwise is treated as a fixed, numeric variable. Defaults to FALSE. Only used in function-based MPMs.

random_indc — A logical value denoting whether to treat individual covariate c as a random, categorical variable. Otherwise is treated as a fixed, numeric variable. Defaults to FALSE. Only used in function-based MPMs.

negfec — A logical value denoting whether fecundity values estimated to be negative should be reset to 0. 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. Only used in function-based MPMs.

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. Only used in function-based MPMs.

censor	If TRUE, then data will be removed according to the variable set in censorcol, such that only data with censor values equal to censorkeep will remain. Defaults to FALSE. Only used in raw MPMs.
censorkeep	The value of the censor variable denoting data elements to keep. Defaults to 0. Only used in raw MPMs.
start_age	The age from which to start the matrix. Defaults to NULL, in which case age 1 is used if prebreeding = TRUE, and age 0 is used if prebreeding = FALSE. Only used in age-based MPMs.
last_age	The final age to use in the matrix. Defaults to NULL, in which case the highest age in the dataset is used. Only used in age-based and age-by-stage MPMs.
fecage_min	The minimum age at which reproduction is possible. Defaults to NULL, which is interpreted to mean that fecundity should be assessed starting in the minimum age observed in the dataset. Only used in age-based MPMs.
fecage_max	The maximum age at which reproduction is possible. Defaults to NULL, which is interpreted to mean that fecundity should be assessed until the final observed age. Only used in age-based MPMs.
fectime	An integer indicating whether to estimate fecundity using the variable given for fec in time t (2) or time t+1 (3). Only used for purely age-based MPMs. Defaults to 2.
fecmod	A scalar multiplier for fecundity. Only used for purely age-based MPMs. Defaults to 1.0.
cont	A logical value designating whether to allow continued survival of individuals past the final age noted in age-based and age-by-stage MPMs, using the demographic characteristics of the final age. Defaults to TRUE.
prebreeding	A logical value indicating whether the life history model is a pre-breeding model. Defaults to TRUE.
stage_NRasRep	A logical value indicating whether to treat non-reproductive individuals as reproductive. Used only in raw, stage-based MPMs in cases where stage assignment must still be handled. Not used in function-based MPMs, and in stage-based MPMs in which a valid hfvdata class data frame with stages already assigned is provided.
initial_nan	A single logical value indicating whether to initialize matrices was all elements set to NaN. Defaults to FALSE. Can only be used with raw MPMs not in sparse format, and should not be used to create normal MPMs intended for projection.
sparse_output	A logical value indicating whether to output matrices in sparse format. Defaults to FALSE, in which case all matrices are output in standard matrix format.

Value

The dominant output is an object of class lefkoMat. If data of class hfv_list for empirical models, or modelsuites of class lefkoModList for function-based models are provided, then a list of class lefkoMatList is provided. The latter is a list in which each element is a separate lefkoMat object, providing output for a bootstrapped MPM analysis.

Class lefkoMat objects are lists holds one full matrix projection model and all of its metadata. The structure has the following elements:

A	A list of full projection matrices in order of sorted patches and occasion times. All matrices output in R's matrix class, or in the dgCMatrix class from the Matrix package if sparse.
U	A list of survival transition matrices sorted as in A. All matrices output in R's matrix class, or in the dgCMatrix class from the Matrix package if sparse.
F	A list of fecundity matrices sorted as in A. All matrices output in R's matrix class, or in the dgCMatrix class from the Matrix package if sparse.
hstages	A data frame matrix showing the pairing of ahistorical stages used to create historical stage pairs. Only used in historical MPMs.
agestages	A data frame showing age-stage pairs. Only used in age-by-stage MPMs.
ahstages	A data frame detailing the characteristics of associated ahistorical stages, in the form of a modified stageframe that includes status as an entry stage through reproduction. Used in all stage-based and age-by-stage MPMs.
labels	A data frame giving the population, patch, and year of each matrix in order.
dataqc	A vector showing the numbers of individuals and rows in the vertical dataset used as input.
matrixqc	A short vector describing the number of non-zero elements in U and F matrices, and the number of annual matrices.
modelqc	This is the qc portion of the modelsuite input.
prob_out	An optional element only added if err_check = TRUE. This is a list of vital rate probability matrices, with 7 columns in the order of survival, observation probability, reproduction probability, primary size transition probability, secondary size transition probability, tertiary size transition probability, and probability of juvenile transition to maturity.
allstages	An optional element only added if err_check = TRUE. This is a data frame giving the values used to determine each matrix element capable of being estimated.
data	An optional element only added if err_check = TRUE and a raw MPM is requested. This consists of the original dataset as edited by this function for indexing purposes.

General Notes

This function automatically determines whether to create a raw or function-based MPM given inputs supplied by the user.

If used, the reproduction matrix (field repmatrix) may be supplied as either historical or ahistorical. If provided as historical, then a historical MPM must be estimated.

If neither a supplement nor a reproduction matrix are used, and the MPM to create is stage-based, then fecundity will be assumed to occur from all reproductive stages to all propagule and immature stages.

Function-based MPM Notes

Users may at times wish to estimate MPMs using a dataset incorporating multiple patches or subpopulations, but without discriminating between those patches or subpopulations. Should the

aim of analysis be a general MPM that does not distinguish these patches or subpopulations, the `modelsearch()` run should not include patch terms.

Input options including multiple variable names must be entered in the order of variables in occasion *t*+1, *t*, and *t*-1. Rearranging the order will lead to erroneous calculations, and will may lead to fatal errors.

This function provides two different means of estimating the probability of size transition. The midpoint method (`CDF = FALSE`) refers to the method in which the probability is estimated by first estimating the probability associated with transition from the exact size at the midpoint of the size class using the corresponding probability density function, and then multiplying that value by the bin width of the size class. Doak et al. 2021 (Ecological Monographs) noted that this method can produce biased results, with total size transitions associated with a specific size not totaling to 1.0 and even specific size transition probabilities capable of being estimated at values greater than 1.0. The alternative and default method (`CDF = TRUE`) uses the cumulative density function to estimate the probability of size transition as the cumulative probability of size transition at the greater limit of the size class minus the cumulative probability of size transition at the lower limit of the size class. This latter method avoids this bias. Note, however, that both methods are exact and unbiased for negative binomial and Poisson distributions.

Under the Gaussian and gamma size distributions, the number of estimated parameters may differ between the two `ipm_method` settings. Because the midpoint method has a tendency to incorporate upward bias in the estimation of size transition probabilities, it is more likely to yield non- zero values when the true probability is extremely close to 0. This will result in the `summary.lefkoMat()` function yielding higher numbers of estimated parameters than the `ipm_method = "CDF"` yields in some cases.

Examples

```
data(lathyrus)

sizevector <- c(0, 4.6, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 1, 2, 3, 4, 5, 6, 7, 8,
  9)
stagevector <- c("Sd", "Sdl", "Dorm", "Sz1nr", "Sz2nr", "Sz3nr", "Sz4nr",
  "Sz5nr", "Sz6nr", "Sz7nr", "Sz8nr", "Sz9nr", "Sz1r", "Sz2r", "Sz3r",
  "Sz4r", "Sz5r", "Sz6r", "Sz7r", "Sz8r", "Sz9r")
repvector <- c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1)
obsvector <- c(0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
matvector <- c(0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0)
indataset <- c(0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 4.6, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
  0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5)

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

lathvertln <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
```

```
  juvcol = "Seedling1988", sizeacol = "lnVol88", repstracol = "Intactseed88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframeln, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, NAas0 = TRUE, censor = TRUE)

lathvertln$feca2 <- round(lathvertln$feca2)
lathvertln$feca1 <- round(lathvertln$feca1)
lathvertln$feca3 <- round(lathvertln$feca3)

lathvertln_adults <- subset(lathvertln, stage2index > 2)
surv_model <- glm(alive3 ~ sizea2 + sizea1 + as.factor(patchid) +
  as.factor(year2), data = lathvertln_adults, family = "binomial")

obs_data <- subset(lathvertln_adults, alive3 == 1)
obs_model <- glm(obsstatus3 ~ as.factor(patchid), data = obs_data,
  family = "binomial")

size_data <- subset(obs_data, obsstatus3 == 1)
siz_model <- lm(sizea3 ~ sizea2 + sizea1 + repstatus1 + as.factor(patchid) +
  as.factor(year2), data = size_data)

reps_model <- glm(repstatus3 ~ sizea2 + sizea1 + as.factor(patchid) +
  as.factor(year2), data = size_data, family = "binomial")

fec_data <- subset(lathvertln_adults, repstatus2 == 1)
fec_model <- glm(feca2 ~ sizea2 + sizea1 + repstatus1 + as.factor(patchid),
  data = fec_data, family = "poisson")

lathvertln_juvs <- subset(lathvertln, stage2index < 3)
jsurv_model <- glm(alive3 ~ as.factor(patchid), data = lathvertln_juvs,
  family = "binomial")

jobs_data <- subset(lathvertln_juvs, alive3 == 1)
jobs_model <- glm(obsstatus3 ~ 1, family = "binomial", data = jobs_data)

jsize_data <- subset(jobs_data, obsstatus3 == 1)
jsiz_model <- lm(sizea3 ~ as.factor(year2), data = jsize_data)

jrepst_model <- 0
jmatst_model <- 1

mod_params <- create_pm(name_terms = TRUE)
mod_params$modelparams[3] <- "patchid"
mod_params$modelparams[4] <- "alive3"
mod_params$modelparams[5] <- "obsstatus3"
mod_params$modelparams[6] <- "sizea3"
mod_params$modelparams[9] <- "repstatus3"
mod_params$modelparams[11] <- "feca2"
mod_params$modelparams[12] <- "sizea2"
mod_params$modelparams[13] <- "sizea1"
mod_params$modelparams[18] <- "repstatus2"
mod_params$modelparams[19] <- "repstatus1"
```

```
used_models <- list(survival_model = surv_model, observation_model = obs_model,
  size_model = siz_model, sizeb_model = 1, sizec_model = 1,
  repstatus_model = reps_model, fecundity_model = fec_model,
  juv_survival_model = jsurv_model, juv_observation_model = jobs_model,
  juv_size_model = jsiz_model, juv_sizeb_model = 1, juv_sizec_model = 1,
  juv_reproduction_model = 0, juv_maturity_model = 1, paramnames = mod_params)

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "mat", "Sd", "Sdl"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "Sdl", "rep", "rep"),
  stage1 = c("Sd", "rep", "Sd", "rep", "Sd", "mat", "mat"),
  eststage3 = c(NA, NA, NA, NA, "mat", NA, NA),
  eststage2 = c(NA, NA, NA, NA, "Sdl", NA, NA),
  eststage1 = c(NA, NA, NA, NA, "Sdl", NA, NA),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, NA, 0.345, 0.054),
  type = c(1, 1, 1, 1, 1, 3, 3), type_t12 = c(1, 2, 1, 2, 1, 1, 1),
  stageframe = lathframeln, historical = TRUE)

# While we do not use MPMs to initialize f_projections3(), we do use MPMs to
# initialize functions start_input() and density_input().
lathmat3ln <- mpm_create(historical = TRUE, year = "all", patch = "all",
  data = lathvertln, stageframe = lathframeln, supplement = lathsupp3,
  modelsuite = used_models, reduce = FALSE)
```

`overwrite` *Create Overwrite Table for MPM Development*

Description

`overwrite()` returns a data frame describing which particular transitions within an ahistorical or historical projection matrix to overwrite with either given rates and probabilities, or other estimated transitions. This function is now deprecated in favor of function `supplemental()`.

Usage

```
overwrite(
  stage3,
  stage2,
  stage1 = NA,
  eststage3 = NA,
  eststage2 = NA,
  eststage1 = NA,
  givenrate = NA,
  type = NA,
  type_t12 = NA
)
```

Arguments

stage3	The name of the stage in occasion *t*+1 in the transition to be replaced. Abbreviations for groups of stages are also allowed (see Notes).
stage2	The name of the stage in occasion *t* in the transition to be replaced. Abbreviations for groups of stages are also allowed (see Notes).
stage1	The name of the stage in occasion *t*-1 in the transition to be replaced. Only needed if a historical matrix is to be produced. Abbreviations for groups of stages are also allowed (see Notes).
eststage3	The name of the stage to replace stage3. Only needed if a transition will be replaced by another estimated transition.
eststage2	The name of the stage to replace stage2. Only needed if a transition will be replaced by another estimated transition.
eststage1	The name of the stage to replace stage1. Only needed if a transition will be replaced by another estimated transition, and the matrix to be estimated is historical.
givenrate	A fixed rate or probability to replace for the transition described by stage3, stage2, and stage1.
type	A 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; or 2, F, or f for the replacement of a fecundity transition. If empty or not provided, then defaults to 1 for survival transition.
type_t12	An optional 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.

Value

A data frame that puts the above vectors together and can be used as input in flefko3(), flefko2(), rlefko3(),rlefko2(), and aflefko2().

Variables in this data frame include the following:

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.
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.
givenrate	A constant to be used as the value of the transition.

convtype — Designates whether the transition from occasion *t* to occasion *t*+1 is a survival-transition probability (1) or a fecundity rate (2).

convtype_t12 — Designates whether the transition from occasion *t*-1 to occasion *t* is a survival transition probability (1), a fecundity rate (2).

Notes

This function is deprecated. Please use `supplemental()`.

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, and leave empty or use `all` if all stages in stageframe are to be used.

Examples

```
cypover2r <- overwrite(stage3 = c("SD", "P1", "P2", "P3", "SL", "D",
    "XSm", "Sm"),
  stage2 = c("SD", "SD", "P1", "P2", "P3", "SL", "SL", "SL"),
  eststage3 = c(NA, NA, NA, NA, NA, "D", "XSm", "Sm"),
  eststage2 = c(NA, NA, NA, NA, NA, "XSm", "XSm", "XSm"),
  givenrate = c(0.1, 0.2, 0.2, 0.2, 0.25, NA, NA, NA),
  type = c("S", "S", "S", "S", "S", "S", "S", "S"))

cypover2r

cypover3r <- overwrite(stage3 = c("SD", "SD", "P1", "P1", "P2", "P3", "SL",
    "D", "XSm", "Sm", "D", "XSm", "Sm"),
  stage2 = c("SD", "SD", "SD", "SD", "P1", "P2", "P3", "SL", "SL", "SL",
    "SL", "SL", "SL"),
  stage1 = c("SD", "rep", "SD", "rep", "SD", "P1", "P2", "P3", "P3", "P3",
    "SL", "SL", "SL"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, NA, "D", "XSm", "Sm", "D", "XSm",
    "Sm"),
  eststage2 = c(NA, NA, NA, NA, NA, NA, NA, "XSm", "XSm", "XSm", "XSm",
    "XSm", "XSm"),
  eststage1 = c(NA, NA, NA, NA, NA, NA, NA, "XSm", "XSm", "XSm", "XSm",
    "XSm", "XSm"),
  givenrate = c(0.1, 0.1, 0.2, 0.2, 0.2, 0.2, 0.25, NA, NA, NA, NA, NA, NA),
  type = c("S", "S", "S", "S", "S", "S", "S", "S", "S", "S", "S", "S", "S"))

cypover3r
```

`plot.lefkoProj` *Plot Projection Simulations*

Description

Function `plot.lefkoProj()` produces plots of `lefkoProj` objects. Acts as a convenient wrapper for the `plot.default()` function.

Usage

```
## S3 method for class 'lefkoProj'
plot(
  x,
  variable = "popsize",
  style = "time",
  repl = "all",
  patch = "pop",
  auto_ylim = TRUE,
  auto_col = TRUE,
  auto_lty = TRUE,
  auto_title = FALSE,
  ...
)
```

Arguments

`x`	A `lefkoProj` object.
`variable`	The focus variable of the plot to produce. Defaults to `"popsize"`, which produces line plots of the `popsize` element in object `x`.
`style`	A string denoting ther kind of plot to produce. Currently limited to `"timeseries"`, which shows `variable` against time on the x axis. Other choices include `"statespace"`, which plots `variable` at one time on the x axis against the same variable in the next time on the y axis.
`repl`	The replicate to plot. Defaults to `"all"`, in which case all replicates are plotted.
`patch`	The patch to plot, as labeled in the `labels` element in object `x`. Defaults to `"pop"`, in which case only the final population-level projection is plotted. Can also be set to `"all"`, in which case projections for all patches and population in the `labels` element are 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 `projection3()` run.

Notes

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

The default settings will preferentially plot any projections marked as `0` in the `patch` portion of the `labels` element of the input MPM. This can produce confusing results if a mean MPM resulting from the `lmean()` function is used as input and the `add_mean` setting is set to the default, which is `TRUE`.

Examples

```
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
matvector <- c(0, 0, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)

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

lathvert <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "Volume88", repstracol = "FCODE88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframe, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, censor = TRUE)

lathrepm <- matrix(0, 7, 7)
lathrepm[1, 6] <- 0.345
lathrepm[2, 6] <- 0.054

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "Sd", "Sdl"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "rep", "rep"),
  stage1 = c("Sd", "rep", "Sd", "rep", "all", "all"),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA),
  multiplier = c(NA, NA, NA, NA, 0.345, 0.054),
  type = c(1, 1, 1, 1, 3, 3), type_t12 = c(1, 2, 1, 2, 1, 1),
  stageframe = lathframe, historical = TRUE)

ehrlen3 <- rlefko3(data = lathvert, stageframe = lathframe,
  year = c(1989, 1990), stages = c("stage3", "stage2", "stage1"),
```

```
  repmatrix = lathrepm, supplement = lathsupp3, yearcol = "year2",
  indivcol = "individ")

lathproj <- projection3(ehrlen3, nreps = 5, stochastic = TRUE)
plot(lathproj)
```

`projection3` *Conduct Population Projection Simulations*

Description

Function `projection3()` runs projection simulations. It projects the population and patches forward in time by a user-defined number of occasions. A given set of matrices is utilized and not recreated, although elements may be altered if density dependence is set. Projections may be deterministic or stochastic, and may be density dependent in either case. If deterministic, then projections will be cyclical if matrices exist covering multiple occasions for each population or patch. If stochastic, then annual matrices will be shuffled within patches and populations. Also produces replicates if set.

Usage

```
projection3(
  mpm,
  nreps = 1L,
  times = 10000L,
  historical = FALSE,
  stochastic = FALSE,
  standardize = FALSE,
  growthonly = TRUE,
  integeronly = FALSE,
  substoch = 0L,
  exp_tol = 700,
  sub_warnings = TRUE,
  quiet = FALSE,
  year = NULL,
  start_vec = NULL,
  start_frame = NULL,
  tweights = NULL,
  density = NULL,
  stage_weights = NULL,
  sparse = NULL
)
```

Arguments

`mpm`	A matrix projection model of class `lefkoMat`, a list of class `lefkoMatList` including bootstrapped MPMs, or a list of full matrix projection matrices.

nreps	The number of replicate projections.
times	Number of occasions to iterate per replicate. Defaults to 10,000.
historical	An optional logical value only used if object mpm is a list of matrices, rather than a lefkoMat object. Defaults to FALSE for the former case, and overridden by information supplied in the lefkoMat object for the latter case.
stochastic	A logical value denoting whether to conduct a stochastic projection or a deterministic / cyclical projection.
standardize	A logical value denoting whether to re-standardize the population size to 1.0 at each occasion. Defaults to FALSE.
growthonly	A logical value indicating whether to produce only the projected population size at each occasion, or a vector showing the stage distribution followed by the reproductive value vector followed by the full population size at each occasion. Defaults to TRUE.
integeronly	A logical value indicating whether to round the number of individuals projected in each stage at each occasion to the nearest integer. Defaults to FALSE.
substoch	An integer value indicating whether to force survival- transition matrices to be substochastic in density dependent simulations. Defaults to 0, which does not force 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.
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.
sub_warnings	A logical value indicating whether to warn the user if density dependence yields matrix values outside of the realm of possibility. Generally, this means that survival-transition elements altered to values outside of the interval [0, 1], and negative fecundity values, will both yield warnings. Defaults to TRUE, but becomes FALSE if quiet = TRUE.
quiet	A logical value indicating whether to suppress warnings. Defaults to FALSE.
year	Either a single integer value corresponding to the year to project, or a vector of times elements with the year to use at each time step. If a vector shorter than times is supplied, then this vector will be cycled. If not provided, then all annual matrices will be cycled within patches or populations.
start_vec	An optional numeric vector denoting the starting stage distribution for the projection. Defaults to a single individual of each stage.
start_frame	An optional data frame characterizing stages, age-stages, or stage-pairs that should be set to non-zero values in the starting vector, and what those values should be. Can only be used with lefkoMat objects.
tweights	An optional numeric vector or matrix denoting the probabilities of choosing each matrix in a stochastic projection. If a matrix is input, 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 a vector is input, then the choice of annual matrix is assumed to be independent of the current matrix. Defaults to equal weighting among matrices.

density — An optional data frame describing the matrix elements that will be subject to density dependence, and the exact kind of density dependence that they will be subject to, or a list of such objects. The data frame used should be an object of class lefkoDens, which is the output from function density_input().

stage_weights — An optional object of class lefkoEq giving the degree to which individuals in each stage are equivalent to one another, or a list of such objects. May also be a numeric vector (or a list thereof), 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.

sparse — A text string indicating whether to use sparse matrix encoding ("yes") or dense matrix encoding ("no"), if the lefkoMat object input as mpm is composed of standard matrices. Defaults to "auto", in which case sparse matrix encoding is used with standard, square matrices with at least 50 rows and no more than 50% of elements with values greater than zero, or when input lefkoMat objects include matrices of class dgCMatrix.

Value

If a lefkoMat object or a simple list of matrices is used as input, then this function will produce a list of class lefkoProj, which always includes the first three elements of the following, and also includes the remaining elements below when a lefkoMat object is used as input:

projection — A list of lists of matrices showing the total number of individuals per stage per occasion. The first list corresponds to each pop-patch followed by each population. The inner list corresponds to replicates within each pop-patch or population.

stage_dist — A list of lists of the actual stage distribution in each occasion in each replicate in each pop-patch or population. The list order is the same as in projection.

rep_value — A list of lists of the actual reproductive value in each occasion in each replicate in each pop-patch or population. The list order is the same as in projection.

pop_size — A list of matrices showing the total population size in each occasion per replicate (row within matrix) per pop-patch or population (list element).

labels — A data frame showing the order of populations and patches in item projection.

ahstages — The original stageframe used in the study.

hstages — A data frame showing the order of historical stage pairs.

agestages — A data frame showing the order of age-stage pairs.

control — A short vector indicating the number of replicates and the number of occasions projected per replicate.

density — The data frame input under the density option. Only provided if input by the user.

If a lefkoMatList object is entered, then this function will produce a list of class lefkoProjList, in which each element is an object of class lefkoProj.

Notes

Users are encourage to run density dependent projections with `lefkoMat` objects as inputs. Users using simple lists of matrices should first develop a life history model and import matrices using function `create_lM()`. Lists of matrices can still be run in this fashion without the use of a `lefkoMat` object, if the user assumes that the stage name is equal to the associated column number in the associated matrix.

Projections are run both at the patch level and at the population level. Population level estimates will be noted at the end of the data frame with `0` entries for patch designation.

Weightings given in `tweights` do not need to sum to 1. Final weightings used will be based on the proportion per element of the sum of elements in the user-supplied vector.

Starting vectors can be input in one of two ways: 1) as `start_vec` input, which is a vector of numbers of the numbers of individuals in each stage, stage pair, or age-stage, with the length of the vector necessarily as long as there are rows in the matrices of the MPM; or 2) as `start_frame` input, which is a data frame showing only those stages, stage pairs, or age-stages that should begin with more than 0 individuals, and the numbers of individuals that those stages should start with (this object is created using the `start_input()` function). If both are provided, then `start_frame` takes precedence and `start_vec` is ignored. If neither is provided, then `projection3()` automatically assumes that each stage, stage pair, or age-stage begins with a single individual. Importantly, if a `lefkoMat` object is not used, and a list of matrices is provided instead, then `start_frame` cannot be utilized and a full `start_vec` must be provided to conduct a simulation with starting numbers of individuals other than 1 per stage.

The resulting data frames in element `projection` are separated by pop-patch according to the order provided in element `labels`, but the matrices for each element of `projection` have the result of each replicate stacked in order on top of one another without any break or indication. Results for each replicate must be separated using the information provided in elements `control` and the 3 stage descriptor elements.

Density dependent projections are automatically set up if object `density` is input. If this object is not included, then density independent projections will be set up. Note that currently, density dependent projections can only be performed with `lefkoMat` objects.

When running density dependent simulations involving user-set exponents, such as the beta term in the Ricker function and both the alpha and beta terms in the Usher function, values above or below the computer limits may cause unpredictable behavior. Noted odd behavior includes sudden shifts in population size to negative values. This function produces warnings when such values are used, and the values used for warnings may be reset with the `exp_tol` term.

The stage distributions and reproductive values produced are not the asymptotic values as would be given by the standardized right and left eigenvectors associated with the dominant eigenvalue of a matrix, but are vectors describing these values at the specific points in time projected. See equations 14.86 and 14.88 and section 14.4 on Sensitivity and Elasticity Analysis under Environmental Stochasticity in Caswell (2001, Matrix Population Models, Sinauer Associates) for more details.

Consistently positive population growth can quickly lead to population size numbers larger than can be handled computationally. In that circumstance, a continuously rising population size will suddenly become `NaN` for the remainder of the projection.

Users wishing to run a projection of a single patch in a `lefkoMat` object with multiple patches should subset the MPM first to contain only the patch needed. This can be accomplished with the `subset_lM()` function.

Speed can sometimes be increased by shifting from automatic sparse matrix determination to forced dense or sparse matrix projection. This will most likely occur when matrices have between 30 and 300 rows and columns. Defaults work best when matrices are very small and dense, or very large and sparse. Speed can also be maximized by keeping the default setting, `integeronly = TRUE`, since the default behavior is to run each projection (replicate) until either the end, or the population size drops to 0. Setting `integeronly = FALSE` may increase runtime dramatically, since the population size can reach extremely small levels without dropping to 0.

See Also

start_input()

density_input()

f_projection3()

append_lP()

summary.lefkoProj()

plot.lefkoProj()

Examples

```
# Lathyrus example
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
matvector <- c(0, 0, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)

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

lathvert <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "Volume88", repstracol = "FCODE88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframe, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, censor = TRUE)

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "Sd", "Sdl"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "rep", "rep"),
  stage1 = c("Sd", "rep", "Sd", "rep", "all", "all"),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA),
  multiplier = c(NA, NA, NA, NA, 0.345, 0.054),
  type = c(1, 1, 1, 1, 3, 3), type_t12 = c(1, 2, 1, 2, 1, 1),
```

```
  stageframe = lathframe, historical = TRUE)

ehrlen3 <- rlefko3(data = lathvert, stageframe = lathframe,
  year = c(1989, 1990), stages = c("stage3", "stage2", "stage1"),
  supplement = lathsupp3, yearcol = "year2", indivcol = "individ")

lathproj <- projection3(ehrlen3, nreps = 5, stochastic = TRUE)

# Cypripedium example
data(cypdata)

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 = 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)

cypsupp3r <- supplemental(stage3 = c("SD", "SD", "P1", "P1", "P2", "P3", "SL",
    "D", "XSm", "Sm", "D", "XSm", "Sm", "mat", "mat", "mat", "SD", "P1"),
  stage2 = c("SD", "SD", "SD", "SD", "P1", "P2", "P3", "SL", "SL", "SL", "SL",
    "SL", "SL", "D", "XSm", "Sm", "rep", "rep"),
  stage1 = c("SD", "rep", "SD", "rep", "SD", "P1", "P2", "P3", "P3", "P3",
    "SL", "SL", "SL", "SL", "SL", "SL", "mat", "mat"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, NA, "D", "XSm", "Sm", "D", "XSm", "Sm",
    "mat", "mat", "mat", NA, NA),
  eststage2 = c(NA, NA, NA, NA, NA, NA, NA, "XSm", "XSm", "XSm", "XSm", "XSm",
    "XSm", "D", "XSm", "Sm", NA, NA),
  eststage1 = c(NA, NA, NA, NA, NA, NA, NA, "XSm", "XSm", "XSm", "XSm", "XSm",
    "XSm", "XSm", "XSm", "XSm", NA, NA),
  givenrate = c(0.1, 0.1, 0.2, 0.2, 0.2, 0.2, 0.25, NA, NA, NA, NA, NA, NA,
    NA, NA, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
    NA, 0.5, 0.5),
  type = c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3),
  type_t12 = c(1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1),
  stageframe = cypframe_raw, historical = TRUE)
```

```
cypmatrix3r <- rlefko3(data = cypraw_v1, stageframe = cypframe_raw,
  year = "all", patch = "all", stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added", "size1added"),
  supplement = cypsupp3r, yearcol = "year2",
  patchcol = "patchid", indivcol = "individ")

cypstoch <- projection3(cypmatrix3r, nreps = 5, stochastic = TRUE)
```

`pyrola` *Demographic Dataset of* Pyrola japonica *and* Pyrola subaphylla *Populations, in Horizontal Format*

Description

A dataset containing the states and fates of *Pyrola japonica* and *Pyrola subaphylla*, family Ericaceae, from populations in the vicinity of Mt. Bandai, Fukushima Prefecture, Japan, resulting from monitoring that occurred annually between 2015 and 2020.

Usage

```
data(pyrola)
```

Format

A data frame with 454 individuals and 57 variables. Each row corresponds to an unique individual, and each variable from `sprouted.2015` on refers to the state of the individual in a particular year.

species String denoting which of the two species the individual belongs to.

population Integer denoting whcih population the individual belongs to. Synonymous with species in this dataset.

id A numeric variable giving a unique number to each individual within each species. Note that numbers are reused among the two species.

sprouted.2015 A binomial indicating whether the individual had living aboveground tissue observable in the 2015 census.

lvs.num.2015 Number of leaves in 2015.

lvs.lng.2015 Length of largest leaf in 2015.

lvs.wdt.2015 Width of largest leaf in 2015.

inf.num.2015 Number of inflorescences in 2015.

inf.lng.tot.2015 Summed inflorescence length in 2015.

flo.tot.2015 Number of flowers in 2015.

frt.tot.2015 Number of fruits in 2015.

sprouted.2016 A binomial indicating whether the individual had living aboveground tissue observable in the 2016 census.

lvs.num.2016 Number of leaves in 2016.

lvs.lng.2016 Length of largest leaf in 2016.

lvs.wdt.2016 Width of largest leaf in 2016.

inf.num.2016 Number of inflorescences in 2016.

inf.lng.tot.2016 Summed inflorescence length in 2016.

flo.tot.2016 Number of flowers in 2016.

frt.tot.2016 Number of fruits in 2016.

sprouted.2017 A binomial indicating whether the individual had living aboveground tissue observable in the 2017 census.

lvs.num.2017 Number of leaves in 2017.

lvs.lng.2017 Length of largest leaf in 2017.

lvs.wdt.2017 Width of largest leaf in 2017.

inf.num.2017 Number of inflorescences in 2017.

inf.lng.tot.2017 Summed inflorescence length in 2017.

flo.tot.2017 Number of flowers in 2017.

frt.tot.2017 Number of fruits in 2017.

sprouted.2018 A binomial indicating whether the individual had living aboveground tissue observable in the 2018 census.

lvs.num.2018 Number of leaves in 2018.

lvs.lng.2018 Length of largest leaf in 2018.

lvs.wdt.2018 Width of largest leaf in 2018.

inf.num.2018 Number of inflorescences in 2018.

inf.lng.tot.2018 Summed inflorescence length in 2018.

flo.tot.2018 Number of flowers in 2018.

frt.tot.2018 Number of fruits in 2018.

sprouted.2019 A binomial indicating whether the individual had living aboveground tissue observable in the 2019 census.

lvs.num.2019 Number of leaves in 2019.

lvs.lng.2019 Length of largest leaf in 2019.

lvs.wdt.2019 Width of largest leaf in 2019.

inf.num.2019 Number of inflorescences in 2019.

inf.lng.tot.2019 Summed inflorescence length in 2019.

flo.tot.2019 Number of flowers in 2019.

frt.tot.2019 Number of fruits in 2019.

sprouted.2020 A binomial indicating whether the individual had living aboveground tissue observable in the 2020 census.

lvs.num.2020 Number of leaves in 2020.

lvs.lng.2020 Length of largest leaf in 2020.

lvs.wdt.2020 Width of largest leaf in 2020.

inf.num.2020 Number of inflorescences in 2020.

inf.lng.tot.2020 Summed inflorescence length in 2020.

flo.tot.2020 Number of flowers in 2020.

frt.tot.2020 Number of fruits in 2020.

Source

Shefferson, R.P., K. Shutoh, and K. Suetsugu. *In review*. Vegetative dormancy and the evolution of mycoheterotrophy in sister *Pyrola* species. *Journal of Ecology*.

Examples

```
data(pyrola)

pyrola$species <- as.factor(pyrola$species)
pyrola$population <- as.factor(pyrola$population)
jreg <- pyrola[which(pyrola$population == 1),]
stagevec_jp <- c("P1", "Sdl", "Dorm", "V0nr", "V1nr", "V2nr", "V3nr", "V4nr",
  "V0r", "V1r", "V2r", "V3r", "V4r")
sizeavec_jp <- c(0, 0, 0, 0, 1, 2, 3, 7, 0, 1, 2, 3, 7)
sizeahbin_jp <- c(0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 3.5, 0.5, 0.5, 0.5, 0.5,
  3.5)
repvec_jp <- c(0, 0, 0, 0, rep(0, 4), rep(1, 5))
propvec_jp <- c(1, rep(0, 12))
immvec_jp <- c(1, 1, rep(0, 11))
matvec_jp <- c(0, 0, rep(1, 11))
obsvec_jp <- c(0, 0, 0, rep(1, 10))
indata_jp <- c(0, 0, rep(1, 11))
comments_jp <- c("protocorm", "seedling", "dormant adult", "stump", "1lf nr",
  "2lf nr", "3lf nr", "4+lf nr", "0lf r", "1lf r", "2lf r", "3lf r",
  "4+lf r")
jp_frame <- sf_create(sizes = sizeavec_jp, stagenames = stagevec_jp,
  binhalfwidth = sizeahbin_jp, repstatus = repvec_jp, obsstatus = obsvec_jp,
  indataset = indata_jp, propstatus = propvec_jp, immstatus = immvec_jp,
  matstatus = matvec_jp, comments = comments_jp)

jhfv <- verticalize3(data = jreg, noyears = 6, firstyear = 2015,
  individcol = "id", blocksize = 8, sizeacol = "lvs.num.2015",
  obsacol = "sprouted.2015", repstracol = "flo.tot.2015",
  repstrbcol = "frt.tot.2015", fecacol = "flo.tot.2015",
  fecbcol = "frt.tot.2015", NAas0 = TRUE, stagesize = "sizea",
  stageassign = jp_frame)

surv_model <- glm(alive3 ~ sizea2 + as.factor(year2), data = jhfv, family = "binomial")

obs_data <- subset(jhfv, alive3 == 1)
obs_model <- glm(obsstatus3 ~ as.factor(year2), data = obs_data, family = "binomial")

size_data <- subset(obs_data, obsstatus3 == 1)
size_model <- glm(sizea3 ~ sizea2, data = size_data, family = "poisson")
```

```
reps_model <- glm(repstatus3 ~ sizea2, data = size_data, family = "binomial")

fec_data <- subset(jhfv, repstatus2 == 1)
fec_model <- MASS::glm.nb(fec2added ~ 1, data = fec_data)

mod_params <- create_pm(name_terms = TRUE)
mod_params$modelparams[4] <- "alive3"
mod_params$modelparams[5] <- "obsstatus3"
mod_params$modelparams[6] <- "sizea3"
mod_params$modelparams[9] <- "repstatus3"
mod_params$modelparams[11] <- "fec2added"
mod_params$modelparams[12] <- "sizea2"
mod_params$modelparams[18] <- "repstatus2"

jp_germ <- 0.90
jp_supp2 <- supplemental(stage3 = c("Sdl", "Dorm", "V0nr", "V1nr", "P1", "Sdl"),
  stage2 = c("P1", "Sdl", "Sdl", "Sdl", "rep", "rep"),
  eststage3 = c(NA, NA, NA, NA, NA, NA),
  eststage2 = c(NA, NA, NA, NA, NA, NA),
  givenrate = c(0.25, 0.35, 0.10, 0.10, NA, NA), # 0.345, 0.054
  multiplier = c(NA, NA, NA, NA, jp_germ * 0.5, jp_germ * 0.5),
  type = c(1, 1, 1, 1, 3, 3), stageframe = jp_frame, historical = FALSE)

jp_ahmpm <- flefko2(year = "all", stageframe = jp_frame, supplement = jp_supp2,
  paramnames = mod_params, surv_model = surv_model, obs_model = obs_model,
  size_model = size_model, repst_model = reps_model, fec_model = fec_model,
  data = jhfv, err_check = TRUE)

lambda3(jp_ahmpm)
```

`repvalue3` *Estimate Reproductive Value*

Description

`repvalue3()` is a generic function that estimates returns the reproductive values of stages in a population projection matrix or a set of matrices. The specifics of estimation vary with the class of input object. This function is made to handle very large and sparse matrices supplied as `lefkoMat` objects or as individual matrices, and can be used with large historical matrices, IPMs, age x stage matrices, as well as ahistorical matrices.

Usage

```
repvalue3(mats, ...)
```

Arguments

`mats`	A lefkoMat object, a population projection matrix, or a list of population projection matrices for which the reproductive value vector is desired.
`...`	Other parameters.

Value

The value returned depends on the class of the `mats` argument. See related functions for details.

See Also

`repvalue3.lefkoMat()`

`repvalue3.matrix()`

`repvalue3.dgCMatrix()`

`repvalue3.list()`

Examples

```
# Lathyrus deterministic example
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
matvector <- c(0, 0, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)

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

lathvert <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "Volume88", repstracol = "FCODE88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframe, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, censor = TRUE)

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "Sd", "Sdl", "mat"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "rep", "rep", "Sdl"),
  stage1 = c("Sd", "rep", "Sd", "rep", "npr", "npr", "Sd"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, "mat"),
  eststage2 = c(NA, NA, NA, NA, NA, NA, "Sdl"),
  eststage1 = c(NA, NA, NA, NA, NA, NA, "NotAlive"),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, 0.345, 0.054, NA),
  type = c(1, 1, 1, 1, 3, 3, 1), type_t12 = c(1, 2, 1, 2, 1, 1, 1),
  stageframe = lathframe, historical = TRUE)

ehrlen3 <- rlefko3(data = lathvert, stageframe = lathframe, year = "all",
  stages = c("stage3", "stage2", "stage1"), supplement = lathsupp3,
  yearcol = "year2", indivcol = "individ")
```

```
ehrlen3mean <- lmean(ehrlen3)
repvalue3(ehrlen3mean)

# Cypripedium stochastic example
data(cypdata)

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 = 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)

# Here we use supplemental() to provide overwrite and reproductive info
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", patch = "all", stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added"), supplement = cypsupp2r,
  yearcol = "year2", patchcol = "patchid", indivcol = "individ")

repvalue3(cypmatrix2r, stochastic = TRUE)
```

`repvalue3.dgCMatrix` *Estimate Reproductive Value Vector for a Single Population Projection Matrix*

Description

`repvalue3.dgCMatrix()` returns the reproductive values for stages in a sparse population projection matrix. The function makes no assumptions about whether the matrix is ahistorical and simply provides standard reproductive values corresponding to each row, meaning that the overall reproductive values of basic life history stages in a historical matrix are not provided (the `repvalue3.lefkoMat()` function estimates these on the basis of stage description information provided in the `lefkoMat` object used as input in that function).

Usage

```
## S3 method for class 'dgCMatrix'
repvalue3(mats, ...)
```

Arguments

`mats`	A population projection matrix.
`...`	Other parameters.

Value

This function returns a vector data frame characterizing the reproductive values for stages of a population projection matrix. This is given as the left eigenvector associated with largest real part of the dominant eigenvalue, divided by the first non-zero element of the left eigenvector.

Notes

Speed can sometimes be increased by shifting from automatic sparse matrix determination to forced dense or sparse matrix projection. This will most likely occur when matrices have several hundred rows and columns. Defaults work best when matrices are very small and dense, or very large and sparse.

See Also

`repvalue3()`

`repvalue3.lefkoMat()`

`repvalue3.matrix()`

`repvalue3.list()`

Examples

```
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
matvector <- c(0, 0, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)

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

lathvert <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "Volume88", repstracol = "FCODE88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframe, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, censor = TRUE)

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "Sd", "Sdl", "mat"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "rep", "rep", "Sdl"),
  stage1 = c("Sd", "rep", "Sd", "rep", "npr", "npr", "Sd"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, "mat"),
  eststage2 = c(NA, NA, NA, NA, NA, NA, "Sdl"),
  eststage1 = c(NA, NA, NA, NA, NA, NA, "NotAlive"),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, 0.345, 0.054, NA),
  type = c(1, 1, 1, 1, 3, 3, 1), type_t12 = c(1, 2, 1, 2, 1, 1, 1),
  stageframe = lathframe, historical = TRUE)

ehrlen3 <- rlefko3(data = lathvert, stageframe = lathframe, year = "all",
  stages = c("stage3", "stage2", "stage1"), supplement = lathsupp3,
  yearcol = "year2", indivcol = "individ", sparse_output = TRUE)

repvalue3(ehrlen3$A[[1]])
```

`repvalue3.lefkoMat` *Estimate Reproductive Value Vectors of Matrices in a lefkoMat Object*

Description

`repvalue3.lefkoMat()` returns the reproductive values for stages in a set of population projection matrices provided as a `lefkoMat` object. This function can handle large and sparse matrices, and

so can be used with large historical matrices, IPMs, age x stage matrices, as well as ahistorical matrices.

Usage

```
## S3 method for class 'lefkoMat'
repvalue3(
  mats,
  stochastic = FALSE,
  times = 10000,
  tweights = NA,
  seed = NA,
  force_sparse = "auto",
  ...
)
```

Arguments

mats	An object of class lefkoMat object.
stochastic	A logical value indicating whether to use deterministic (FALSE) or stochastic (TRUE) analysis. Defaults to FALSE.
times	An integer variable indicating number of occasions to project if using stochastic analysis. Defaults to 10000.
tweights	An optional numeric vector or matrix denoting the probabilities of choosing each matrix in a stochastic projection. If a matrix is input, 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 a vector is input, then the choice of annual matrix is assumed to be independent of the current matrix. Defaults to equal weighting among matrices.
seed	A number to use as a random number seed.
force_sparse	A text string indicating whether to use sparse matrix encoding ("yes") when supplied with standard matrices. Defaults to "auto", in which case sparse matrix encoding is used with square matrices with at least 50 rows and no more than 50% of elements with values greater than zero.
...	Other parameters.

Value

This function returns the asymptotic reproductive value vectors if deterministic analysis is chosen, and long-run mean reproductive value vectors if stochastic analysis is chosen.

The output depends on whether the lefkoMat object used as input is ahistorical or historical, and whether the analysis is deterministic or stochastic. If deterministic and ahistorical, then a single data frame is output, which includes the number of the matrix within the A element of the input lefkoMat object, followed by the stage id (numeric and assigned through sf_create()), the stage name, and the estimated proportion of the reproductive value vector (rep_value). If stochastic and ahistorical, then a single data frame is output starting with the number of the population-patch (matrix_set), a string concatenating the names of the population and the patch (poppatch), the assigned stage id

number (stage_id), and the stage name (stage), and the long-run mean reproductive value vector (rep_value).

If a historical matrix is used as input, then two data frames are output into a list object. The hist element describes the historical stage-pair reproductive values, while the ahist element describes the stage reproductive values. If deterministic, then hist contains a data frame including the matrix number (matrix), the numeric stage designations for stages in occasions *t* and *t*-1, (stage_id_2 and stage_id_1, respectively), followed by the respective stage names (stage_2 and stage_1), and ending with the estimated reproductive values (rep_value). The associated ahist element is as before. If stochastic, then the hist element contains a single data frame with the number of the population-patch (matrix_set), a string concatenating the names of the population and the patch (poppatch), the assigned stage id numbers in times *t* and *t*-1 (stage_id_2 and stage_id_2, respectively), and the associated stage names (stage_2 and stage_1, respectively), and the long-run mean reproductive values (rep_value). The associated ahist element is as before in the ahistorical, stochastic case.

In addition to the data frames noted above, stochastic analysis will result in the additional output of a list of matrices containing the actual projected reproductive value vectors across all projected occasions, in the order of population-patch combinations in the lefkoMat input.

Notes

In stochastic analysis, the projected mean reproductive value vector is the arithmetic mean across the final projected 1000 occasions if the simulation is at least 2000 projected occasions long. If between 500 and 2000 projected occasions long, then only the final 200 are used, and if fewer than 500 occasions are used, then all are used. Note that because reproductive values in stochastic simulations can change greatly in the initial portion of the run, we encourage a minimum 2000 projected occasions per simulation, with 10000 preferred.

Speed can sometimes be increased by shifting from automatic sparse matrix determination to forced dense or sparse matrix projection. This will most likely occur when matrices have several hundred rows and columns. Defaults work best when matrices are very small and dense, or very large and sparse.

See Also

repvalue3()

repvalue3.matrix()

repvalue3.dgCMatrix()

repvalue3.list()

Examples

```
data(cypdata)

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 = 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)

# Here we use supplemental() to provide overwrite and reproductive info
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", patch = "all", stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added"), supplement = cypsupp2r,
  yearcol = "year2", patchcol = "patchid", indivcol = "individ")

repvalue3(cypmatrix2r, stochastic = TRUE)
```

`repvalue3.lefkoMatList` *Estimate Reproductive Value Vectors of Matrices in a lefkoMatList Object*

Description

`repvalue3.lefkoMatList()` returns the reproductive values for stages in sets of population projection matrices provided within a `lefkoMatList` object. This function can handle large and sparse matrices, and so can be used with large historical matrices, IPMs, age x stage matrices, as well as ahistorical matrices.

Usage

```
## S3 method for class 'lefkoMatList'
repvalue3(
  mats,
  stochastic = FALSE,
  times = 10000,
  tweights = NA,
  seed = NA,
  force_sparse = "auto",
  ...
)
```

Arguments

mats	An object of class lefkoMatList object.
stochastic	A logical value indicating whether to use deterministic (FALSE) or stochastic (TRUE) analysis. Defaults to FALSE.
times	An integer variable indicating number of occasions to project if using stochastic analysis. Defaults to 10000.
tweights	An optional numeric vector or matrix denoting the probabilities of choosing each matrix in a stochastic projection. If a matrix is input, 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 a vector is input, then the choice of annual matrix is assumed to be independent of the current matrix. Defaults to equal weighting among matrices.
seed	A number to use as a random number seed.
force_sparse	A text string indicating whether to use sparse matrix encoding ("yes") when supplied with standard matrices. Defaults to "auto", in which case sparse matrix encoding is used with square matrices with at least 50 rows and no more than 50% of elements with values greater than zero.
...	Other parameters.

Value

This function returns a list with two elements. The first is the mean reproductive value vector (and long-run mean reproductive value vectors from stochastic analysis), and the second is a list of reproductive value vectors (and long-run mean reproductive value vectors from stochastic analysis) corresponding to the lefkoMat objects in the original lefkoMatList list input in argument mats.

The vector format depends on whether the lefkoMat object used as input is ahistorical or historical, and whether the analysis is deterministic or stochastic. If deterministic and ahistorical, then a single data frame is output, which includes the number of the matrix within the A element of the input lefkoMat object, followed by the stage id (numeric and assigned through sf_create()), the stage name, and the estimated proportion of the reproductive value vector (rep_value). If stochastic and ahistorical, then a single data frame is output starting with the number of the population-patch (matrix_set), a string concatenating the names of the population and the patch (poppatch), the assigned stage id number (stage_id), and the stage name (stage), and the long-run mean reproductive value vector (rep_value).

If a historical matrix is used as input, then two data frames are output into a list object. The `hist` element describes the historical stage-pair reproductive values, while the `ahist` element describes the stage reproductive values. If deterministic, then `hist` contains a data frame including the matrix number (`matrix`), the numeric stage designations for stages in occasions *t* and *t*-1, (`stage_id_2` and `stage_id_1`, respectively), followed by the respective stage names (`stage_2` and `stage_1`), and ending with the estimated reproductive values (`rep_value`). The associated `ahist` element is as before. If stochastic, then the `hist` element contains a single data frame with the number of the population-patch (`matrix_set`), a string concatenating the names of the population and the patch (`poppatch`), the assigned stage id numbers in times *t* and *t*-1 (`stage_id_2` and `stage_id_2`, respectively), and the associated stage names (`stage_2` and `stage_1`, respectively), and the long-run mean reproductive values (`rep_value`). The associated `ahist` element is as before in the ahistorical, stochastic case.

In addition to the data frames noted above, stochastic analysis will result in the additional output of a list of matrices containing the actual projected reproductive value vectors across all projected occasions, in the order of population-patch combinations in the `lefkoMat` input.

Notes

In stochastic analysis, the projected mean reproductive value vector is the arithmetic mean across the final projected 1000 occasions if the simulation is at least 2000 projected occasions long. If between 500 and 2000 projected occasions long, then only the final 200 are used, and if fewer than 500 occasions are used, then all are used. Note that because reproductive values in stochastic simulations can change greatly in the initial portion of the run, we encourage a minimum 2000 projected occasions per simulation, with 10000 preferred.

Speed can sometimes be increased by shifting from automatic sparse matrix determination to forced dense or sparse matrix projection. This will most likely occur when matrices have several hundred rows and columns. Defaults work best when matrices are very small and dense, or very large and sparse.

See Also

`repvalue3()`

`repvalue3.matrix()`

`repvalue3.dgCMatrix()`

`repvalue3.list()`

Examples

```
# Lathyrus deterministic example
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
matvector <- c(0, 0, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
```

```
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)

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

lathvert <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "Volume88", repstracol = "FCODE88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframe, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, censor = TRUE)

lathvert_boot <- bootstrap3(lathvert, reps = 3)

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "Sd", "Sdl", "mat"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "rep", "rep", "Sdl"),
  stage1 = c("Sd", "rep", "Sd", "rep", "npr", "npr", "Sd"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, "mat"),
  eststage2 = c(NA, NA, NA, NA, NA, NA, "Sdl"),
  eststage1 = c(NA, NA, NA, NA, NA, NA, "NotAlive"),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, 0.345, 0.054, NA),
  type = c(1, 1, 1, 1, 3, 3, 1), type_t12 = c(1, 2, 1, 2, 1, 1, 1),
  stageframe = lathframe, historical = TRUE)

ehrlen3_boot <- rlefko3(data = lathvert_boot, stageframe = lathframe,
  year = "all", stages = c("stage3", "stage2", "stage1"),
  supplement = lathsupp3, yearcol = "year2", indivcol = "individ")

ehrlen3mean <- lmean(ehrlen3_boot)
repvalue3(ehrlen3mean)

# Cypripedium stochastic example
data(cypdata)

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 = 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)

cypraw_v1_boot <- bootstrap3(cypraw_v1, reps = 3)

# Here we use supplemental() to provide overwrite and reproductive info
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_boot <- rlefko2(data = cypraw_v1_boot, stageframe = cypframe_raw,
  year = "all", patch = "all", stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added"), supplement = cypsupp2r,
  yearcol = "year2", patchcol = "patchid", indivcol = "individ")

repvalue3(cypmatrix2r_boot, stochastic = TRUE)
```

`repvalue3.list` *Estimate Reproductive Value Vector for a List of Projection Matrices*

Description

`repvalue3.list()` returns the reproductive values for stages in population projection matrices arranged in a general list. The function makes no assumptions about whether the matrix is ahistorical and simply provides standard reproductive values corresponding to each row, meaning that the overall reproductive values of basic life history stages in a historical matrix are not provided (the `repvalue3.lefkoMat()` function estimates these on the basis of stage description information provided in the `lefkoMat` object used as input in that function). This function can handle large and sparse matrices, and so can be used with large historical matrices, IPMs, age x stage matrices, as well as smaller ahistorical matrices.

Usage

```
## S3 method for class 'list'
repvalue3(
  mats,
```

```
  stochastic = FALSE,
  times = 10000,
  tweights = NA,
  seed = NA,
  force_sparse = "auto",
  ...
)
```

Arguments

mats	A list of population projection matrices, all in either class matrix or class dgCMatrix.
stochastic	A logical value indicating whether to use deterministic (FALSE) or stochastic (TRUE) analysis. Defaults to FALSE.
times	An integer variable indicating number of occasions to project if using stochastic analysis. Defaults to 10000.
tweights	An optional numeric vector or matrix denoting the probabilities of choosing each matrix in a stochastic projection. If a matrix is input, 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 a vector is input, then the choice of annual matrix is assumed to be independent of the current matrix. Defaults to equal weighting among matrices.
seed	A number to use as a random number seed in stochastic projection.
force_sparse	A text string indicating whether to use sparse matrix encoding ("yes") when supplied with standard matrices. Defaults to "auto", in which case sparse matrix encoding is used with square matrices with at least 50 rows and no more than 50% of elements with values greater than zero.
...	Other parameters.

Value

This function returns a list of vector data frames characterizing the reproductive values for stages of each population projection matrix. This is given as the left eigenvector associated with largest real part of the dominant eigenvalue, divided by the first non-zero element of the left eigenvector.

Notes

Speed can sometimes be increased by shifting from automatic sparse matrix determination to forced dense or sparse matrix projection. This will most likely occur when matrices have several hundred rows and columns. Defaults work best when matrices are very small and dense, or very large and sparse.

See Also

repvalue3()

repvalue3.lefkoMat()

repvalue3.dgCMatrix()

repvalue3.matrix()

Examples

```
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
matvector <- c(0, 0, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)

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

lathvert <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "Volume88", repstracol = "FCODE88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframe, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, censor = TRUE)

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "Sd", "Sdl", "mat"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "rep", "rep", "Sdl"),
  stage1 = c("Sd", "rep", "Sd", "rep", "npr", "npr", "Sd"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, "mat"),
  eststage2 = c(NA, NA, NA, NA, NA, NA, "Sdl"),
  eststage1 = c(NA, NA, NA, NA, NA, NA, "NotAlive"),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, 0.345, 0.054, NA),
  type = c(1, 1, 1, 1, 3, 3, 1), type_t12 = c(1, 2, 1, 2, 1, 1, 1),
  stageframe = lathframe, historical = TRUE)

ehrlen3 <- rlefko3(data = lathvert, stageframe = lathframe, year = "all",
  stages = c("stage3", "stage2", "stage1"), supplement = lathsupp3,
  yearcol = "year2", indivcol = "individ")

repvalue3(ehrlen3$A)
```

`repvalue3.matrix` *Estimate Reproductive Value Vector for a Single Population Projection Matrix*

Description

`repvalue3.matrix()` returns the reproductive values for stages in a population projection matrix. The function makes no assumptions about whether the matrix is ahistorical and simply provides

standard reproductive values corresponding to each row, meaning that the overall reproductive values of basic life history stages in a historical matrix are not provided (the repvalue3.lefkoMat() function estimates these on the basis of stage description information provided in the lefkoMat object used as input in that function). This function can handle large and sparse matrices, and so can be used with large historical matrices, IPMs, age x stage matrices, as well as smaller ahistorical matrices.

Usage

```
## S3 method for class 'matrix'
repvalue3(mats, force_sparse = "auto", ...)
```

Arguments

mats	A population projection matrix.
force_sparse	A text string indicating whether to use sparse matrix encoding ("yes") when supplied with standard matrices. Defaults to "auto", in which case sparse matrix encoding is used with square matrices with at least 50 rows and no more than 50% of elements with values greater than zero.
...	Other parameters.

Value

This function returns a vector data frame characterizing the reproductive values for stages of a population projection matrix. This is given as the left eigenvector associated with largest real part of the dominant eigenvalue, divided by the first non-zero element of the left eigenvector.

Notes

Speed can sometimes be increased by shifting from automatic sparse matrix determination to forced dense or sparse matrix projection. This will most likely occur when matrices have between 30 and 300 rows and columns. Defaults work best when matrices are very small and dense, or very large and sparse.

See Also

repvalue3()

repvalue3.lefkoMat()

repvalue3.dgCMatrix()

repvalue3.list()

Examples

```
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
```

```
matvector <- c(0, 0, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)

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

lathvert <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "Volume88", repstracol = "FCODE88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframe, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, censor = TRUE)

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "Sd", "Sdl", "mat"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "rep", "rep", "Sdl"),
  stage1 = c("Sd", "rep", "Sd", "rep", "npr", "npr", "Sd"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, "mat"),
  eststage2 = c(NA, NA, NA, NA, NA, NA, "Sdl"),
  eststage1 = c(NA, NA, NA, NA, NA, NA, "NotAlive"),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, 0.345, 0.054, NA),
  type = c(1, 1, 1, 1, 3, 3, 1), type_t12 = c(1, 2, 1, 2, 1, 1, 1),
  stageframe = lathframe, historical = TRUE)

ehrlen3 <- rlefko3(data = lathvert, stageframe = lathframe, year = "all",
  stages = c("stage3", "stage2", "stage1"), supplement = lathsupp3,
  yearcol = "year2", indivcol = "individ")

ehrlen3mean <- lmean(ehrlen3)
repvalue3(ehrlen3mean$A[[1]])
```

`ricker3` *Two-Parameter Ricker Function*

Description

Function `ricker3()` creates a vector of values produced by the two- parameter Ricker function as applied with a user-specified time lag. The two-parameter Ricker function is given as $\phi_{t+1} = \phi_t \alpha e^{-\beta n_t}$. Here, if no `separate_N` vector is provided, then $n_t = \phi_t$.

Usage

```
ricker3(
  start_value,
```

```
  alpha,
  beta,
  time_steps = 100L,
  time_lag = 1L,
  pre0_subs = FALSE,
  pre0_value = 0,
  substoch = 0L,
  separate_N = NULL
)
```

Arguments

start_value	A positive number to start the return vector in time 0.
alpha	The alpha parameter in the two-parameter Ricker function. Must be non-negative.
beta	The beta parameter in the two-parameter Ricker function.
time_steps	The number of time steps to run the projection. Must be a positive integer.
time_lag	A positive integer denoting the number of time steps back for the value of phi in the two-parameter Ricker function.
pre0_subs	A logical value indicating whether to use a number other than that given in start_value for values of phi lagged from times prior to time 0.
pre0_value	A positive number to use for phi lagged from times prior to time 0. Only used if pre0_subs = TRUE.
substoch	An integer value indicating the kind of substochasticity to use. Values include: 0, no substochasticity enforced (the default); 1, all numbers must be non-negative; and 2, all numbers should be forced to the interval [0, 1].
separate_N	An optional numeric vector with values of N in each time, if phi is to be treated as different from N in the two-parameter model.

Value

A numeric vector of values showing values projected under the two- parameter Ricker function.

Examples

```
trial_run1 <- ricker3(1, alpha = 0.5, beta = -0.009)
plot(trial_run1)

trial_run2 <- ricker3(1, alpha = 0.5, beta = 0.009)
plot(trial_run2)

trial_run3 <- ricker3(1, alpha = 1, beta = -0.009)
plot(trial_run3)

trial_run4 <- ricker3(1, alpha = 1, beta = 0.009)
plot(trial_run4)

trial_run5 <- ricker3(1, alpha = 5, beta = -0.009)
plot(trial_run5)
```

```
trial_run6 <- ricker3(1, alpha = 5, beta = 0.009)
plot(trial_run6)

used_Ns <- c(10, 15, 12, 14, 14, 150, 15, 1, 5, 7, 9, 14, 13, 16, 17, 19,
  25, 26)
trial_run7 <- ricker3(1, alpha = 1, beta = -0.009, separate_N = used_Ns)
plot(trial_run7)
```

`rlefko2` *Create Raw Ahistorical Matrix Projection Model*

Description

Function `rlefko2()` returns raw ahistorical MPMs, including the associated component transition and fecundity matrices, a data frame describing the ahistorical stages used, and a data frame describing the population, patch, and occasion time associated with each matrix.

Usage

```
rlefko2(
  data,
  stageframe,
  year = "all",
  pop = NULL,
  patch = NULL,
  censor = FALSE,
  stages = NULL,
  alive = c("alive3", "alive2"),
  obsst = NULL,
  size = c("sizea3", "sizea2"),
  sizeb = NULL,
  sizec = NULL,
  repst = c("repstatus3", "repstatus2"),
  matst = c("matstatus3", "matstatus2"),
  fec = c("feca3", "feca2"),
  supplement = NULL,
  repmatrix = NULL,
  overwrite = NULL,
  yearcol = NULL,
  popcol = NULL,
  patchcol = NULL,
  indivcol = NULL,
  censorcol = NULL,
  censorkeep = 0,
  NRasRep = FALSE,
  reduce = FALSE,
```

```
  simple = FALSE,
  err_check = FALSE,
  initial_nan = FALSE,
  sparse_output = FALSE
)
```

Arguments

data	A vertical demographic data frame, with variables corresponding to the naming conventions in functions verticalize3() and historicalize3(). Alternatively, a list of bootstrapped data of class hfv_list.
stageframe	A stageframe object that includes information on the size, observation status, propagule status, reproduction status, immaturity status, and maturity status of each ahistorical stage.
year	A variable corresponding to observation occasion, or a set of such values, given in values associated with the year term used in vital rate model development. Can also equal "all", in which case matrices will be estimated for all occasion times. Defaults to "all".
pop	A variable designating which populations will have matrices estimated. Should be set to specific population names, or to "all" if all populations should have matrices estimated.
patch	A variable designating which patches or subpopulations will have matrices estimated. Should be set to specific patch names, or to "all" if matrices should be estimated for all patches. Defaults to NA, in which case patch designations are ignored..
censor	If TRUE, then data will be removed according to the variable set in censorcol, such that only data with censor values equal to censorkeep will remain. Defaults to FALSE.
stages	An optional vector denoting the names of the variables within the main vertical dataset coding for the stages of each individual in occasions t+1 and t. The names of stages in these variables should match those used in the stageframe exactly. If left blank, then rlefko2() will attempt to infer stages by matching values of alive, size, repst, and matst to characteristics noted in the associated stageframe.
alive	A vector of names of binomial variables corresponding to status as alive (1) or dead (0) in occasions t+1 and t, respectively.
obsst	A vector of names of binomial variables corresponding to observation status in occasions t+1, t, and t-1, respectively. Defaults to NULL, in which case observation status is not used.
size	A vector of names of variables coding the primary size variable in occasions t+1 and t, respectively. Defaults to c("sizea3", "sizea2").
sizeb	A vector of names of variables coding the secondary size variable in occasions t+1 and t, respectively. Defaults to NULL, in which case this variable is not used.
sizec	A vector of names of variables coding the tertiary size variable in occasions t+1 and t, respectively. Defaults to NULL, in which case this variable is not used.

`repst` A vector of names of variables coding reproductive status in occasions *t*+1 and *t*, respectively. Defaults to `c("repstatus3", "repstatus2")`. Must be supplied if `stages` is not provided.

`matst` A vector of names of variables coding maturity status in occasions *t*+1 and *t*, respectively. Defaults to `c("matstatus3", "matstatus2")`. Must be supplied if `stages` is not provided.

`fec` A vector of names of variables coding fecundity in occasions *t*+1 and *t*, respectively. Defaults to `c("feca3", "feca2")`.

`supplement` An optional data frame of class `lefkoSD` that provides supplemental data that should be incorporated into the MPM. Three kinds of data may be integrated this way: transitions to be estimated via the use of proxy transitions, transition overwrites from the literature or supplemental studies, and transition multipliers for fecundity. This data frame should be produced using the `supplemental()` function. Should be used in place of or in addition to an overwrite table (see `overwrite` below) and a reproduction matrix (see `repmatrix` below).

`repmatrix` An optional reproduction matrix. This matrix is composed mostly of 0s, with non-zero entries acting as element identifiers and multipliers for fecundity (with 1 equaling full fecundity). If left blank, and no `supplement` is provided, then `rlefko2()` will assume that all stages marked as reproductive produce offspring at 1x that of estimated fecundity, and that offspring production will yield the first stage noted as propagule or immature. To prevent this behavior, input just `0`, which will result in fecundity being estimated only for transitions noted in `supplement` above. Must be the dimensions of an ahistorical matrix.

`overwrite` An optional data frame developed with the `overwrite()` function describing transitions to be overwritten either with given values or with other estimated transitions. Note that this function supplements overwrite data provided in `supplement`.

`yearcol` The variable name or column number corresponding to occasion *t* in the dataset.

`popcol` The variable name or column number corresponding to the identity of the population.

`patchcol` The variable name or column number corresponding to patch in the dataset.

`indivcol` The variable name or column number coding individual identity.

`censorcol` The variable name or column number denoting the censor status. Only needed if `censor = TRUE`.

`censorkeep` The value of the censor variable denoting data elements to keep. Defaults to `0`.

`NRasRep` If `data` does not include stage assignments, then this option determines whether non-reproductive and reproductive individuals should be lumped into the same stages. Defaults to `FALSE`.

`reduce` A logical value denoting whether to remove ahistorical stages associated with only zero transitions. These are removed only if the respective row and column sums in ALL matrices estimated equal 0. Defaults to `FALSE`.

`simple` A logical value indicating whether to produce `A`, `U`, and `F` matrices, or only the latter two. Defaults to `FALSE`, in which case all three are output.

`err_check` A logical value indicating whether to append extra information used in matrix calculation within the output list. Defaults to `FALSE`.

initial_nan	A single logical value indicating whether to initialize matrices was all elements set to NaN. Defaults to FALSE. Cannot be used with sparse matrices.
sparse_output	A logical value indicating whether to output matrices in sparse format. Defaults to FALSE, in which case all matrices are output in standard matrix format.

Value

If the user inputs a standard hfv_data object in argument data, then this function will return an object of class lefkoMat. If the user inputs an object of class hfv_list in argument data, then the output will be an object of class lefkoMatList, in which each element is an object of class lefkoMat.

A lefkoMat object is a list that holds one full matrix projection model and all of its metadata. The structure has the following elements:

A	A list of full projection matrices in order of sorted populations, patches, and occasions. All matrices output in the matrix class, or in the dgCMatrix class from the Matrix package if sparse.
U	A list of survival transition matrices sorted as in A. All matrices output in the matrix class, or in the dgCMatrix class from the Matrix package if sparse.
F	A list of fecundity matrices sorted as in A. All matrices output in the matrix class, or in the dgCMatrix class from the Matrix package if sparse.
hstages	A data frame matrix showing the pairing of ahistorical stages used to create historical stage pairs. Set to NA for ahistorical matrices.
agestages	A data frame showing age-stage pairs. In this function, it is set to NA. Only used in output to function aflefko2().
ahstages	A data frame detailing the characteristics of associated ahistorical stages, in the form of a modified stageframe that includes status as an entry stage through reproduction.
labels	A data frame giving the population, patch, and year of each matrix in order.
dataqc	A vector showing the numbers of individuals and rows in the vertical dataset used as input.
matrixqc	A short vector describing the number of non-zero elements in U and F matrices, and the number of annual matrices.
modelqc	This is the qc portion of the modelsuite input in function-based MPMs. Empty in this function.

Notes

The default behavior of this function is to estimate fecundity with regards to transitions specified via associated fecundity multipliers in the supplement. If this field is left empty, then fecundity will be estimated at full for all transitions leading from reproductive stages to immature and propagule stages.

Users may at times wish to estimate MPMs using a dataset incorporating multiple patches or subpopulations. Should the aim of analysis be a general MPM that does not distinguish these patches or subpopulations, the patchcol variable should be left to NA, which is the default. Otherwise the variable identifying patch needs to be named.

Input options including multiple variable names must be entered in the order of variables in occasion t+1 and t. Rearranging the order WILL lead to erroneous calculations, and may lead to fatal errors.

Although this function is capable of assigning stages given an input stageframe, it lacks the power of verticalize3() and historicalize3() in this regard. Users are strongly encouraged to use the latter two functions for stage assignment.

See Also

mpm_create()

flefko3()

flefko2()

aflefko2()

arlefko2()

fleslie()

rlefko3()

rleslie()

Examples

```
# Lathyrus example
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
matvector <- c(0, 0, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)

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

lathvert <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "Volume88", repstracol = "FCODE88",
  fecacol = "Intactseed88", deadacol = "Dead1988", nonobsacol = "Dormant1988",
  stageassign = lathframe, stagesize = "sizea", censorcol = "Missing1988",
  censorkeep = NA, censor = TRUE)

lathsupp2 <- supplemental(stage3 = c("Sd", "Sdl", "Sd", "Sdl"),
  stage2 = c("Sd", "Sd", "rep", "rep"),
  givenrate = c(0.345, 0.054, NA, NA),
  multiplier = c(NA, NA, 0.345, 0.054),
  type = c(1, 1, 3, 3), stageframe = lathframe, historical = FALSE)
```

```
ehrlen2 <- rlefko2(data = lathvert, stageframe = lathframe, year = "all",
  stages = c("stage3", "stage2"), supplement = lathsupp2, yearcol = "year2",
  indivcol = "individ")

# Cypripedium example
data(cypdata)

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 = 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)

# Here we use supplemental() to provide overwrite and reproductive info
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", patch = "all", stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added"), supplement = cypsupp2r,
  yearcol = "year2", patchcol = "patchid", indivcol = "individ")
```

---

rlefko3 *Create Raw Historical Matrix Projection Model*

Description

Function `rlefko3()` returns raw historical MPMs, including the associated component transition and fecundity matrices, data frames describing the ahistorical stages used and the historical paired stages, and a data frame describing the population, patch, and occasion time associated with each matrix.

Usage

```
rlefko3(
  data,
  stageframe,
  year = "all",
  pop = NULL,
  patch = NULL,
  censor = FALSE,
  stages = NULL,
  alive = c("alive3", "alive2", "alive1"),
  obsst = NULL,
  size = c("sizea3", "sizea2", "sizea1"),
  sizeb = NULL,
  sizec = NULL,
  repst = c("repstatus3", "repstatus2", "repstatus1"),
  matst = c("matstatus3", "matstatus2", "matstatus1"),
  fec = c("feca3", "feca2", "feca1"),
  supplement = NULL,
  repmatrix = NULL,
  overwrite = NULL,
  yearcol = NULL,
  popcol = NULL,
  patchcol = NULL,
  indivcol = NULL,
  censorcol = NULL,
  censorkeep = 0,
  NRasRep = FALSE,
  format = "ehrlen",
  reduce = FALSE,
  simple = FALSE,
  err_check = FALSE,
  initial_nan = FALSE,
  sparse_output = FALSE
)
```

Arguments

`data`	A vertical demographic data frame, with variables corresponding to the naming conventions in functions `verticalize3()` and `historicalize3()`. Alternatively, a list of bootstrapped data of class `hfv_list`.

`stageframe` A stageframe object that includes information on the size, observation status, propagule status, reproduction status, immaturity status, and maturity status of each ahistorical stage.

`year` A variable corresponding to observation occasion, or a set of such values, given in values associated with the `year` term used in vital rate model development. Can also equal `"all"`, in which case matrices will be estimated for all occasions. Defaults to `"all"`.

`pop` A variable designating which populations will have matrices estimated. Should be set to specific population names, or to `"all"` if all populations should have matrices estimated.

`patch` A variable designating which patches or subpopulations will have matrices estimated. Should be set to specific patch names, or to `"all"` if matrices should be estimated for all patches. Defaults to `NA`, in which case patch designations are ignored..

`censor` If `TRUE`, then data will be removed according to the variable set in `censorcol`, such that only data with censor values equal to `censorkeep` will remain. Defaults to `FALSE`.

`stages` An optional vector denoting the names of the variables within the main vertical dataset coding for the stages of each individual in occasions *t*+1, *t*, and *t*-1. The names of stages in these variables should match those used in the `stageframe` exactly. If left blank, then `rlefko3()` will attempt to infer stages by matching values of `alive`, `size`, `repst`, and `matst` to characteristics noted in the associated `stageframe`.

`alive` A vector of names of binomial variables corresponding to status as alive (`1`) or dead (`0`) in occasions *t*+1, *t*, and *t*-1, respectively.

`obsst` A vector of names of binomial variables corresponding to observation status in occasions *t*+1, *t*, and *t*-1, respectively. Defaults to `NULL`, in which case observation status is not used.

`size` A vector of names of variables coding the primary size variable in occasions *t*+1, *t*, and *t*-1, respectively. Defaults to `c("sizea3", "sizea2", "sizea1")`.

`sizeb` A vector of names of variables coding the secondary size variable in occasions *t*+1, *t*, and *t*-1, respectively. Defaults to `NULL`, in which case this variable is not used.

`sizec` A vector of names of variables coding the tertiary size variable in occasions *t*+1, *t*, and *t*-1, respectively. Defaults to `NULL`, in which case this variable is not used.

`repst` A vector of names of variables coding reproductive status in occasions *t*+1, *t*, and *t*-1, respectively. Defaults to `c("repstatus3", "repstatus2", "repstatus1")`. Must be supplied if `stages` is not provided.

`matst` A vector of names of variables coding maturity status in occasions *t*+1, *t*, and *t*-1, respectively. Defaults to `c("matstatus3", "matstatus2", "matstatus1")`. Must be supplied if `stages` is not provided.

`fec` A vector of names of variables coding fecundity in occasions *t*+1, *t*, and *t*-1, respectively. Defaults to `c("feca3", "feca2", "feca1")`.

`supplement` An optional data frame of class `lefkoSD` that provides supplemental data that should be incorporated into the MPM. Three kinds of data may be integrated this way: transitions to be estimated via the use of proxy transitions, transition overwrites from the literature or supplemental studies, and transition multipliers for fecundity. This data frame should be produced using the `supplemental()` function. Should be used in place of or in addition to an overwrite table (see `overwrite` below) and a reproduction matrix (see `repmatrix` below).

`repmatrix` An optional reproduction matrix. This matrix is composed mostly of 0s, with non-zero entries acting as element identifiers and multipliers for fecundity (with 1 equaling full fecundity). If left blank, and no `supplement` is provided, then `rlefko3()` will assume that all stages marked as reproductive produce offspring at 1x that of estimated fecundity, and that offspring production will yield the first stage noted as propagule or immature. To prevent this behavior, input just `0`, which will result in fecundity being estimated only for transitions noted in `supplement` above. May be the dimensions of either a historical or an ahistorical matrix. If the former, then the fecundity estimation of this function may be unpredictable. If the latter, then all stages will be used in occasion *t*-1 for each suggested ahistorical transition.

`overwrite` An optional data frame developed with the `overwrite()` function describing transitions to be overwritten either with given values or with other estimated transitions. Note that this function supplements overwrite data provided in `supplement`.

`yearcol` The variable name or column number corresponding to occasion *t* in the dataset.

`popcol` The variable name or column number corresponding to the identity of the population.

`patchcol` The variable name or column number corresponding to patch in the dataset.

`indivcol` The variable name or column number coding individual identity.

`censorcol` The variable name or column number denoting the censor status. Only needed if `censor = TRUE`.

`censorkeep` The value of the censor variable denoting data elements to keep. Defaults to `0`.

`NRasRep` If `data` does not include stage assignments, then this option determines whether non-reproductive and reproductive individuals should be lumped into the same stages. Defaults to `FALSE`.

`format` A string indicating whether to estimate matrices in `ehrlen` format or `deVries` format. The latter adds one unborn prior stage to account for the prior state of newborns. Defaults to `ehrlen` format.

`reduce` A logical value denoting whether to remove historical stages associated exclusively with zero transitions. These are removed only if the respective row and column sums in ALL matrices estimated equal 0. Defaults to `FALSE`.

`simple` A logical value indicating whether to produce `A`, `U`, and `F` matrices, or only the latter two. Defaults to `FALSE`, in which case all three are output.

`err_check` A logical value indicating whether to append extra information used in matrix calculation within the output list. Defaults to `FALSE`.

`initial_nan` A single logical value indicating whether to initialize matrices was all elements set to `NaN`. Defaults to `FALSE`. Cannot be used with sparse matrices.

sparse_output A logical value indicating whether to output matrices in sparse format. Defaults to FALSE, in which case all matrices are output in standard matrix format.

Value

If the user inputs a standard hfv_data object in argument data, then this function will return an object of class lefkoMat. If the user inputs an object of class hfv_list in argument data, then the output will be an object of class lefkoMatList, in which each element is an object of class lefkoMat.

A lefkoMat object is a list that holds one full matrix projection model and all of its metadata. The structure has the following elements:

A A list of full projection matrices in order of sorted populations, patches, and occasions. All matrices output in the matrix class, or in the dgCMatrix class from the Matrix package if sparse.

U A list of survival transition matrices sorted as in A. All matrices output in the matrix class, or in the dgCMatrix class from the Matrix package if sparse.

F A list of fecundity matrices sorted as in A. All matrices output in the matrix class, or in the dgCMatrix class from the Matrix package if sparse.

hstages A data frame matrix showing the pairing of ahistorical stages used to create historical stage pairs.

agestages A data frame showing age-stage pairs. In this function, it is set to NA. Only used in output to function aflefko2().

ahstages A data frame detailing the characteristics of associated ahistorical stages, in the form of a modified stageframe that includes status as an entry stage through reproduction.

labels A data frame giving the population, patch, and year of each matrix in order.

dataqc A vector showing the numbers of individuals and rows in the vertical dataset used as input.

matrixqc A short vector describing the number of non-zero elements in U and F matrices, and the number of annual matrices.

modelqc This is the qc portion of the modelsuite input in function-based MPMs. Empty in this function.

Notes

The default behavior of this function is to estimate fecundity with regards to transitions specified via associated fecundity multipliers in the supplement. If this field is left empty, then fecundity will be estimated at full for all transitions leading from reproductive stages to immature and propagule stages.

Users may at times wish to estimate MPMs using a dataset incorporating multiple patches or subpopulations. Should the aim of analysis be a general MPM that does not distinguish these patches or subpopulations, the patchcol variable should be left to NA, which is the default. Otherwise the variable identifying patch needs to be named.

Input options including multiple variable names must be entered in the order of variables in occasion *t*+1, *t*, and *t*-1. Rearranging the order WILL lead to erroneous calculations, and may lead to fatal errors.

Although this function is capable of assigning stages given an input stageframe, it lacks the power of verticalize3() and historicalize3() in this regard. Users are strongly encouraged to use the latter two functions for stage assignment.

See Also

mpm_create()

flefko3()

flefko2()

aflefko2()

arlefko2()

fleslie()

rlefko2()

rleslie()

Examples

```
# Lathyrus example
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
matvector <- c(0, 0, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)

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

lathvert <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "Volume88", repstracol = "FCODE88",
  fecacol = "Intactseed88", deadacol = "Dead1988", nonobsacol = "Dormant1988",
  stageassign = lathframe, stagesize = "sizea", censorcol = "Missing1988",
  censorkeep = NA, censor = TRUE)

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "Sd", "Sdl", "mat"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "rep", "rep", "Sdl"),
  stage1 = c("Sd", "rep", "Sd", "rep", "npr", "npr", "Sd"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, "mat"),
```

```
  eststage2 = c(NA, NA, NA, NA, NA, NA, "Sdl"),
  eststage1 = c(NA, NA, NA, NA, NA, NA, "NotAlive"),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, 0.345, 0.054, NA),
  type = c(1, 1, 1, 1, 3, 3, 1), type_t12 = c(1, 2, 1, 2, 1, 1, 1),
  stageframe = lathframe, historical = TRUE)

ehrlen3 <- rlefko3(data = lathvert, stageframe = lathframe, year = "all",
  stages = c("stage3", "stage2", "stage1"), supplement = lathsupp3,
  yearcol = "year2", indivcol = "individ")

# Cypripedium example
data(cypdata)

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 = 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)

cypsupp3r <- supplemental(stage3 = c("SD", "SD", "P1", "P1", "P2", "P3", "SL",
    "D", "XSm", "Sm", "D", "XSm", "Sm", "mat", "mat", "mat", "SD", "P1"),
  stage2 = c("SD", "SD", "SD", "SD", "P1", "P2", "P3", "SL", "SL", "SL", "SL",
    "SL", "SL", "D", "XSm", "Sm", "rep", "rep"),
  stage1 = c("SD", "rep", "SD", "rep", "SD", "P1", "P2", "P3", "P3", "P3",
    "SL", "SL", "SL", "SL", "SL", "SL", "mat", "mat"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, NA, "D", "XSm", "Sm", "D", "XSm", "Sm",
    "mat", "mat", "mat", NA, NA),
  eststage2 = c(NA, NA, NA, NA, NA, NA, NA, "XSm", "XSm", "XSm", "XSm", "XSm",
    "XSm", "D", "XSm", "Sm", NA, NA),
  eststage1 = c(NA, NA, NA, NA, NA, NA, NA, "XSm", "XSm", "XSm", "XSm", "XSm",
    "XSm", "XSm", "XSm", "XSm", NA, NA),
  givenrate = c(0.1, 0.1, 0.2, 0.2, 0.2, 0.2, 0.25, NA, NA, NA, NA, NA, NA,
    NA, NA, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
    NA, 0.5, 0.5),
```

```
  type = c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3),
  type_t12 = c(1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1),
  stageframe = cypframe_raw, historical = TRUE)

cypmatrix3r <- rlefko3(data = cypraw_v1, stageframe = cypframe_raw,
  year = "all", patch = "all", stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added", "size1added"),
  supplement = cypsupp3r, yearcol = "year2", patchcol = "patchid",
  indivcol = "individ")
```

`rleslie` *Create Raw Leslie (Age-based) Matrix Projection Model*

Description

Function `rleslie()` returns raw Leslie MPMs, including the associated component transition and fecundity matrices, a data frame describing the ages used, and a data frame describing the population, patch, and occasion time associated with each matrix.

Usage

```
rleslie(
  data,
  start_age = NA,
  last_age = NA,
  continue = TRUE,
  fecage_min = NA,
  fecage_max = NA,
  alive = c("alive3", "alive2", "alive1"),
  repst = c("repstatus3", "repstatus2", "repstatus1"),
  fec = c("feca3", "feca2", "feca1"),
  agecol = "obsage",
  year = "all",
  supplement = NULL,
  pop = NULL,
  patch = NULL,
  yearcol = NULL,
  popcol = NULL,
  patchcol = NULL,
  indivcol = NULL,
  censor = FALSE,
  censorcol = NULL,
  censorkeep = 0,
  fectime = 2,
  fecmod = 1,
  prebreeding = TRUE,
  reduce = FALSE,
```

```
  simple = FALSE,
  err_check = FALSE,
  initial_nan = FALSE,
  sparse_output = FALSE
)
```

Arguments

data	A vertical demographic data frame, with variables corresponding to the naming conventions in functions verticalize3() and historicalize3(). Alternatively, a list of bootstrapped data of class hfv_list.
start_age	The age from which to start the matrix. Defaults to NA, age 1 is used if prebreeding = TRUE, and age 0 is used if prebreeding = FALSE.
last_age	The final age to use in the matrix. Defaults to NA, in which case the highest age in the dataset is used.
continue	A logical value designating whether to allow continued survival of individuals past the final age noted in the stageframe, using the demographic characteristics of the final age. Defaults to TRUE.
fecage_min	The minimum age at which reproduction is possible. Defaults to NA, which is interpreted to mean that fecundity should be assessed starting in the minimum age observed in the dataset.
fecage_max	The maximum age at which reproduction is possible. Defaults to NA, which is interpreted to mean that fecundity should be assessed until the final observed age.
alive	A vector of names of binomial variables corresponding to status as alive (1) or dead (0) in occasions *t*+1 ans *t*, respectively.
repst	A vector of names of variables coding reproductive status in occasions *t*+1 and *t*, respectively. Defaults to c("repstatus3", "repstatus2").
fec	A vector of names of variables coding fecundity in occasions *t*+1 and *t*, respectively. Defaults to c("feca3", "feca2").
agecol	The name or column number of the variable coding for age in data. Defaults to "obsage".
year	A variable corresponding to observation occasion, or a set of such values, given in values associated with the year term used in vital rate model development. Can also equal "all", in which case matrices will be estimated for all occasion times. Defaults to "all".
supplement	An optional data frame of class lefkoSD that provides supplemental data that should be incorporated into the MPM. Three kinds of data may be integrated this way: transitions to be estimated via the use of proxy transitions, transition overwrites from the literature or supplemental studies, and transition multipliers for survival and fecundity. This data frame should be produced using the supplemental() function.
pop	A variable designating which populations will have matrices estimated. Should be set to specific population names, or to "all" if all populations should have matrices estimated.

patch	A variable designating which patches or subpopulations will have matrices estimated. Should be set to specific patch names, or to "all" if matrices should be estimated for all patches. Defaults to "all".
yearcol	The variable name or column number corresponding to occasion *t* in the dataset.
popcol	The variable name or column number corresponding to the identity of the population.
patchcol	The variable name or column number corresponding to patch in the dataset.
indivcol	The variable name or column number coding individual identity.
censor	If TRUE, then data will be removed according to the variable set in censorcol, such that only data with censor values equal to censorkeep will remain. Defaults to FALSE.
censorcol	The variable name or column number denoting the censor status. Only needed if censor = TRUE.
censorkeep	The value of the censor variable denoting data elements to keep. Defaults to 0.
fectime	An integer indicating whether to estimate fecundity using the variable given for fec in time *t* (2) or time *t*+1 (3).
fecmod	A scalar multiplier for fecundity. Defaults to 1.0.
prebreeding	A logical value indicating whether the life history model is a pre-breeding model. Defaults to TRUE.
reduce	A logical value denoting whether to remove ages associated with only zero transitions. These are removed only if the respective row and column sums in ALL matrices estimated equal 0. Defaults to FALSE, and should generally not be used in age-based MPMs.
simple	A logical value indicating whether to produce A, U, and F matrices, or only the latter two. Defaults to FALSE, in which case all three are output.
err_check	A logical value indicating whether to append extra information used in matrix calculation within the output list. Defaults to FALSE.
initial_nan	A single logical value indicating whether to initialize matrices was all elements set to NaN. Defaults to FALSE. Cannot be used with sparse matrices.
sparse_output	A logical value indicating whether to output matrices in sparse format. Defaults to FALSE, in which case all matrices are output in standard matrix format.

Value

If the user inputs a standard hfv_data object in argument data, then this function will return an object of class lefkoMat. If the user inputs an object of class hfv_list in argument data, then the output will be an object of class lefkoMatList, in which each element is an object of class lefkoMat.

A lefkoMat object is a list that holds one full matrix projection model and all of its metadata. The structure has the following elements:

A	A list of full projection matrices in order of sorted populations, patches, and occasions. All matrices output in the matrix class, or in the dgCMatrix class from the Matrix package if sparse.

U	A list of survival transition matrices sorted as in A. All matrices output in the matrix class, or in the dgCMatrix class from the Matrix package if sparse.
F	A list of fecundity matrices sorted as in A. All matrices output in the matrix class, or in the dgCMatrix class from the Matrix package if sparse.
hstages	A data frame matrix showing the pairing of ahistorical stages used to create historical stage pairs. Set to NA for ahistorical matrices.
agestages	A data frame showing age-stage pairs. In this function, it is set to NA. Only used in output to function aflefko2().
ahstages	A data frame detailing the characteristics of associated ahistorical stages, in the form of a modified stageframe that includes status as an entry stage through reproduction.
labels	A data frame giving the population, patch, and year of each matrix in order.
dataqc	A vector showing the numbers of individuals and rows in the vertical dataset used as input.
matrixqc	A short vector describing the number of non-zero elements in U and F matrices, and the number of annual matrices.
modelqc	This is the qc portion of the modelsuite input in function-based MPMs. Empty in this function.

Notes

In order to accomodate survival to time t+1 in the final year of a study, the maximum age assessed if no input last_age is provided is one time step past the final described age.

Users may at times wish to estimate MPMs using a dataset incorporating multiple patches or subpopulations. Should the aim of analysis be a general MPM that does not distinguish these patches or subpopulations, the patchcol variable should be left to NA, which is the default. Otherwise the variable identifying patch needs to be named.

Input options including multiple variable names must be entered in the order of variables in occasion t+1 and t. Rearranging the order WILL lead to erroneous calculations, and may lead to fatal errors.

See Also

mpm_create()

flefko3()

flefko2()

aflefko2()

arlefko2()

fleslie()

rlefko3()

rlefko2()

Examples

```
data(cypdata)

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",
  age_offset = 3, NAas0 = TRUE, NRasRep = TRUE)

cyp_rl <- rleslie(data = cypraw_v1, start_age = 0, last_age = 4,
  continue = TRUE, fecage_min = 3, year = "all", pop = NA, patch = "all",
  yearcol = "year2", patchcol = "patchid", indivcol = "individ")
```

---

sensitivity3 *Estimate Sensitivity of Population Growth Rate to Matrix Elements*

Description

`sensitivity3()` is a generic function that returns the sensitivity of the population growth rate to the elements of the matrices in a matrix population model. Currently, this function estimates both deterministic and stochastic sensitivities, where the growth rate is λ in the former case and the log of the stochastic λ in the latter case. This function is made to handle very large and sparse matrices supplied as `lefkoMat` objects, as lists of matrices, and as individual matrices.

Usage

```
sensitivity3(mats, ...)
```

Arguments

`mats`	A lefkoMat object, or population projection matrix, for which the stable stage distribution is desired.
`...`	Other parameters

Value

The value returned depends on the class of the `mats` argument.

See Also

`sensitivity3.lefkoMat()`

`sensitivity3.matrix()`

`sensitivity3.dgCMatrix()`

`sensitivity3.list()`

Examples

```
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
matvector <- c(0, 0, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)

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

lathvert <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "Volume88", repstracol = "FCODE88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframe, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, censor = TRUE)

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "Sd", "Sdl", "mat"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "rep", "rep", "Sdl"),
  stage1 = c("Sd", "rep", "Sd", "rep", "npr", "npr", "Sd"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, "mat"),
  eststage2 = c(NA, NA, NA, NA, NA, NA, "Sdl"),
  eststage1 = c(NA, NA, NA, NA, NA, NA, "NotAlive"),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, 0.345, 0.054, NA),
  type = c(1, 1, 1, 1, 3, 3, 1), type_t12 = c(1, 2, 1, 2, 1, 1, 1),
  stageframe = lathframe, historical = TRUE)

ehrlen3 <- rlefko3(data = lathvert, stageframe = lathframe, year = "all",
  stages = c("stage3", "stage2", "stage1"), supplement = lathsupp3,
  yearcol = "year2", indivcol = "individ")

sensitivity3(ehrlen3)
```

`sensitivity3.dgCMatrix` *Estimate Sensitivity of Population Growth Rate of a Single Matrix*

Description

`sensitivity3.dgCMatrix()` returns the sensitivities of λ to elements of a single, sparse matrix. Because this handles only one matrix, sensitivities are inherently deterministic and based on the dominant eigen value as the best metric of the population growth rate.

Usage

```
## S3 method for class 'dgCMatrix'
sensitivity3(mats, sparse = "auto", ...)
```

Arguments

`mats`	An object of class `dgCMatrix`.
`sparse`	A text string indicating whether to use sparse matrix encoding (`"yes"`) or dense matrix encoding (`"no"`). Defaults to `"auto"`, in which case sparse matrix encoding is used with square matrices with at least 50 rows and no more than 50% of elements with values greater than zero.
`...`	Other parameters.

Value

This function returns a single deterministic sensitivity matrix.

Notes

All sensitivity matrix outputs from this function are in standard matrix format.

See Also

`sensitivity3()`

`sensitivity3.lefkoMat()`

`sensitivity3.list()`

`sensitivity3.matrix()`

`sensitivity3.lefkoMatList()`

Examples

```
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
matvector <- c(0, 0, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)
```

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

lathvert <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "Volume88", repstracol = "FCODE88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframe, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, censor = TRUE)

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "Sd", "Sdl", "mat"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "rep", "rep", "Sdl"),
  stage1 = c("Sd", "rep", "Sd", "rep", "npr", "npr", "Sd"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, "mat"),
  eststage2 = c(NA, NA, NA, NA, NA, NA, "Sdl"),
  eststage1 = c(NA, NA, NA, NA, NA, NA, "NotAlive"),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, 0.345, 0.054, NA),
  type = c(1, 1, 1, 1, 3, 3, 1), type_t12 = c(1, 2, 1, 2, 1, 1, 1),
  stageframe = lathframe, historical = TRUE)

ehrlen3 <- rlefko3(data = lathvert, stageframe = lathframe, year = "all",
  stages = c("stage3", "stage2", "stage1"), supplement = lathsupp3,
  yearcol = "year2", indivcol = "individ", sparse_output = TRUE)

sensitivity3(ehrlen3$A[[1]])
```

`sensitivity3.lefkoMat` *Estimate Sensitivity of Population Growth Rate of a lefkoMat Object*

Description

`sensitivity3.lefkoMat()` returns the sensitivities of population growth rate to elements of all `$A` matrices in an object of class `lefkoMat`. If deterministic, then λ is taken as the population growth rate. If stochastic, then the log of stochastic λ, or the log stochastic growth rate, is taken as the population growth rate. This function can handle large and sparse matrices, and so can be used with large historical matrices, IPMs, age x stage matrices, as well as smaller ahistorical matrices.

Usage

```
## S3 method for class 'lefkoMat'
sensitivity3(
  mats,
  stochastic = FALSE,
  times = 10000,
```

```
  tweights = NA,
  seed = NA,
  sparse = "auto",
  append_mats = FALSE,
  ...
)
```

Arguments

mats	An object of class lefkoMat.
stochastic	A logical value determining whether to conduct a deterministic (FALSE) or stochastic (TRUE) sensitivity analysis. Defaults to FALSE.
times	The number of occasions to project forward in stochastic simulation. Defaults to 10000.
tweights	An optional numeric vector or matrix denoting the probabilities of choosing each matrix in a stochastic projection. If a matrix is input, 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 a vector is input, then the choice of annual matrix is assumed to be independent of the current matrix. Defaults to equal weighting among matrices.
seed	A number to use as a random number seed in stochastic projection.
sparse	A text string indicating whether to use sparse matrix encoding ("yes") or dense matrix encoding ("no"). Defaults to "auto", in which case sparse matrix encoding is used with square matrices with at least 50 rows and no more than 50% of elements with values greater than zero.
append_mats	A logical value indicating whether to include the original A, U, and F matrices in the output lefkoSens object.
...	Other parameters.

Value

This function returns an object of class lefkoSens, which is a list of 8 elements. The first, h_sensmats, is a list of historical sensitivity matrices (NULL if an ahMPM is used as input). The second, ah_elasmats, is a list of either ahistorical sensitivity matrices if an ahMPM is used as input, or, if an hMPM is used as input, then the result is a list of ahistorical matrices based on the equivalent historical dependencies assumed in the input historical matrices. The third element, hstages, is a data frame showing historical stage pairs (NULL if an ahMPM used as input). The fourth element, agestages, show the order of age-stage combinations, if age-by-stage MPMs have been supplied. The fifth element, ahstages, is a data frame showing the order of ahistorical stages. The last 3 elements are the A, U, and F portions of the input.

Notes

All sensitivity matrix outputs from this function are in standard matrix format.

Deterministic sensitivities are estimated as eqn. 9.14 in Caswell (2001, Matrix Population Models). Stochastic sensitivities are estimated as eqn. 14.97 in Caswell (2001). Note that stochastic sensitivities are of the log of the stochastic λ.

Speed can sometimes be increased by shifting from automatic sparse matrix determination to forced dense or sparse matrix projection. This will most likely occur when matrices have between 30 and 300 rows and columns. Defaults work best when matrices are very small and dense, or very large and sparse.

The `time_weights` and `steps` arguments are now deprecated. Instead, please use the `tweights` and `times` arguments.

See Also

`sensitivity3()`

`sensitivity3.matrix()`

`sensitivity3.dgCMatrix()`

`sensitivity3.list()`

`sensitivity3.lefkoMatList()`

Examples

```
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
matvector <- c(0, 0, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)

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

lathvert <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "Volume88", repstracol = "FCODE88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframe, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, censor = TRUE)

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "Sd", "Sdl", "mat"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "rep", "rep", "Sdl"),
  stage1 = c("Sd", "rep", "Sd", "rep", "npr", "npr", "Sd"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, "mat"),
  eststage2 = c(NA, NA, NA, NA, NA, NA, "Sdl"),
  eststage1 = c(NA, NA, NA, NA, NA, NA, "NotAlive"),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, 0.345, 0.054, NA),
  type = c(1, 1, 1, 1, 3, 3, 1), type_t12 = c(1, 2, 1, 2, 1, 1, 1),
```

```
  stageframe = lathframe, historical = TRUE)

ehrlen3 <- rlefko3(data = lathvert, stageframe = lathframe, year = "all",
  stages = c("stage3", "stage2", "stage1"), supplement = lathsupp3,
  yearcol = "year2", indivcol = "individ")

sensitivity3(ehrlen3, stochastic = TRUE)
```

`sensitivity3.lefkoMatList` *Estimate Sensitivity of Population Growth Rate of a lefkoMatList Object*

Description

`sensitivity3.lefkoMatList()` returns the sensitivities of population growth rate to elements of all `$A` matrices in all `lefkoMat` objects in a list of class `lefkoMatList`. If deterministic, then λ is taken as the population growth rate. If stochastic, then the log of stochastic λ, or the log stochastic growth rate, is taken as the population growth rate. This function can handle large and sparse matrices, and so can be used with large historical matrices, IPMs, age x stage matrices, as well as smaller ahistorical matrices.

Usage

```
## S3 method for class 'lefkoMatList'
sensitivity3(
  mats,
  stochastic = FALSE,
  times = 10000,
  tweights = NA,
  seed = NA,
  sparse = "auto",
  append_mats = FALSE,
  ...
)
```

Arguments

`mats`	An object of class `lefkoMatList`.
`stochastic`	A logical value determining whether to conduct a deterministic (FALSE) or stochastic (TRUE) sensitivity analysis. Defaults to FALSE.
`times`	The number of occasions to project forward in stochastic simulation. Defaults to `10000`.
`tweights`	An optional numeric vector or matrix denoting the probabilities of choosing each matrix in a stochastic projection. If a matrix is input, 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 a vector is input, then the choice of annual matrix is assumed to be independent of the current matrix. Defaults to equal weighting among matrices.
`seed`	A number to use as a random number seed in stochastic projection.
`sparse`	A text string indicating whether to use sparse matrix encoding (`"yes"`) or dense matrix encoding (`"no"`). Defaults to `"auto"`, in which case sparse matrix encoding is used with square matrices with at least 50 rows and no more than 50% of elements with values greater than zero.
`append_mats`	A logical value indicating whether to include the original A, U, and F matrices in the output `lefkoSens` object.
`...`	Other parameters.

Value

This function returns a list with two elements. The first is an object of class `lefkoSens` that contains the mean sensitivity matrices of the bootstrapped sensitivity matrices. The second is a list containing `lefkoSens` objects giving the sensitivity matrices of all bootstrapped matrices.

Within these lists are objects of class `lefkoSens`, which are comprised of lists of 8 elements. The first, `h_sensmats`, is a list of historical sensitivity matrices (`NULL` if an ahMPM is used as input). The second, `ah_elasmats`, is a list of either ahistorical sensitivity matrices if an ahMPM is used as input, or, if an hMPM is used as input, then the result is a list of ahistorical matrices based on the equivalent historical dependencies assumed in the input historical matrices. The third element, `hstages`, is a data frame showing historical stage pairs (`NULL` if an ahMPM used as input). The fourth element, `agestages`, show the order of age-stage combinations, if age-by-stage MPMs have been supplied. The fifth element, `ahstages`, is a data frame showing the order of ahistorical stages. The last 3 elements are the A, U, and F portions of the input.

Notes

All sensitivity matrix outputs from this function are in standard matrix format.

Deterministic sensitivities are estimated as eqn. 9.14 in Caswell (2001, Matrix Population Models). Stochastic sensitivities are estimated as eqn. 14.97 in Caswell (2001). Note that stochastic sensitivities are of the log of the stochastic λ.

Speed can sometimes be increased by shifting from automatic sparse matrix determination to forced dense or sparse matrix projection. This will most likely occur when matrices have between 30 and 300 rows and columns. Defaults work best when matrices are very small and dense, or very large and sparse.

The `time_weights` and `steps` arguments are now deprecated. Instead, please use the `tweights` and `times` arguments.

See Also

`sensitivity3()`

`sensitivity3.lefkoMat()`

`sensitivity3.matrix()`

sensitivity3.dgCMatrix()

sensitivity3.list()

Examples

```
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
matvector <- c(0, 0, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)

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

lathvert <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "Volume88", repstracol = "FCODE88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframe, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, censor = TRUE)

lathvert_boot <- bootstrap3(lathvert, reps = 3)

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "Sd", "Sdl", "mat"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "rep", "rep", "Sdl"),
  stage1 = c("Sd", "rep", "Sd", "rep", "npr", "npr", "Sd"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, "mat"),
  eststage2 = c(NA, NA, NA, NA, NA, NA, "Sdl"),
  eststage1 = c(NA, NA, NA, NA, NA, NA, "NotAlive"),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, 0.345, 0.054, NA),
  type = c(1, 1, 1, 1, 3, 3, 1), type_t12 = c(1, 2, 1, 2, 1, 1, 1),
  stageframe = lathframe, historical = TRUE)

ehrlen3_boot <- rlefko3(data = lathvert_boot, stageframe = lathframe,
  year = "all", stages = c("stage3", "stage2", "stage1"),
  supplement = lathsupp3, yearcol = "year2", indivcol = "individ")

sensitivity3(ehrlen3_boot, stochastic = TRUE)
```

sensitivity3.list *Estimate Sensitivity of Population Growth Rate of a List of Matrices*

Description

sensitivity3.list() returns the sensitivities of population growth rate to elements of matrices supplied in a list. The sensitivity analysis can be deterministic or stochastic, but if the latter then at least two A matrices must be included in the list. This function can handle large and sparse matrices, and so can be used with large historical matrices, IPMs, age x stage matrices, as well as smaller ahistorical matrices.

Usage

```
## S3 method for class 'list'
sensitivity3(
  mats,
  stochastic = FALSE,
  times = 10000,
  tweights = NA,
  historical = FALSE,
  seed = NA,
  sparse = "auto",
  append_mats = FALSE,
  ...
)
```

Arguments

mats	An object of class matrix.
stochastic	A logical value determining whether to conduct a deterministic (FALSE) or stochastic (TRUE) sensitivity analysis. Defaults to FALSE.
times	The number of occasions to project forward in stochastic simulation. Defaults to 10,000.
tweights	An optional numeric vector or matrix denoting the probabilities of choosing each matrix in a stochastic projection. If a matrix is input, 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 a vector is input, then the choice of annual matrix is assumed to be independent of the current matrix. Defaults to equal weighting among matrices.
historical	A logical value indicating whether matrices are historical. Defaults to FALSE.
seed	A number to use as a random number seed in stochastic projection.
sparse	A text string indicating whether to use sparse matrix encoding ("yes") or dense matrix encoding ("no"). Defaults to "auto", in which case sparse matrix encoding is used with square matrices with at least 50 rows and no more than 50% of elements with values greater than zero.

append_mats	A logical value indicating whether to include the original matrices input as object mats in the output lefkoSense object. Defaults to FALSE.
...	Other parameters.

Value

This function returns an object of class lefkoSens, which is a list of 8 elements. The first, h_sensmats, is a list of historical sensitivity matrices (NULL if an ahMPM is used as input). The second, ah_elasmats, is a list of ahistorical sensitivity matrices if an ahMPM is used as input (NULL if an hMPM is used as input). The third element, hstages, the fourth element, agestages, and the fifth element, ahstages, are NULL. The last 3 elements include the original A matrices supplied (as the A element), followed by NULLs for the U and F elements.

Notes

All sensitivity matrix outputs from this function are in standard matrix format.

Deterministic sensitivities are estimated as eqn. 9.14 in Caswell (2001, Matrix Population Models). Stochastic sensitivities are estimated as eqn. 14.97 in Caswell (2001). Note that stochastic sensitivities are with regard to the log of the stochastic λ.

Currently, this function does not estimate equivalent ahistorical stochastic sensitivities for input historical matrices, due to the lack of guidance input on the order of stages (guidance is provided within lefkoMat objects).

Speed can sometimes be increased by shifting from automatic sparse matrix determination to forced dense or sparse matrix projection. This will most likely occur when matrices have between 30 and 300 rows and columns. Defaults work best when matrices are very small and dense, or very large and sparse.

The time_weights and steps arguments are now deprecated. Instead, please use the tweights and times arguments.

See Also

sensitivity3()

sensitivity3.lefkoMat()

sensitivity3.matrix()

sensitivity3.dgCMatrix()

sensitivity3.lefkoMatList()

Examples

```
# Lathyrus example
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
matvector <- c(0, 0, 1, 1, 1, 1, 1)
```

```
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)

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

lathvert <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "Volume88", repstracol = "FCODE88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframe, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, censor = TRUE)

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "Sd", "Sdl", "mat"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "rep", "rep", "Sdl"),
  stage1 = c("Sd", "rep", "Sd", "rep", "npr", "npr", "Sd"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, "mat"),
  eststage2 = c(NA, NA, NA, NA, NA, NA, "Sdl"),
  eststage1 = c(NA, NA, NA, NA, NA, NA, "NotAlive"),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, 0.345, 0.054, NA),
  type = c(1, 1, 1, 1, 3, 3, 1), type_t12 = c(1, 2, 1, 2, 1, 1, 1),
  stageframe = lathframe, historical = TRUE)

ehrlen3 <- rlefko3(data = lathvert, stageframe = lathframe, year = "all",
  stages = c("stage3", "stage2", "stage1"), supplement = lathsupp3,
  yearcol = "year2", indivcol = "individ")

sensitivity3(ehrlen3$A)

# Cypripedium example
data(cypdata)

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 = 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", "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", patch = "all", stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added"), supplement = cypsupp2r,
  yearcol = "year2", patchcol = "patchid", indivcol = "individ")

sensitivity3(cypmatrix2r$A)
```

`sensitivity3.matrix` *Estimate Sensitivity of Population Growth Rate of a Single Matrix*

Description

`sensitivity3.matrix()` returns the sensitivities of λ to elements of a single matrix. Because this handles only one matrix, the sensitivities are inherently deterministic and based on the dominant eigen value as the best metric of the population growth rate. This function can handle large and sparse matrices, and so can be used with large historical matrices, IPMs, age x stage matrices, as well as smaller ahistorical matrices.

Usage

```
## S3 method for class 'matrix'
sensitivity3(mats, sparse = "auto", ...)
```

Arguments

`mats`	An object of class `matrix`.
`sparse`	A text string indicating whether to use sparse matrix encoding (`"yes"`) or dense matrix encoding (`"no"`). Defaults to `"auto"`, in which case sparse matrix encoding is used with square matrices with at least 50 rows and no more than 50% of elements with values greater than zero.
`...`	Other parameters.

Value

This function returns a single deterministic sensitivity matrix.

Notes

All sensitivity matrix outputs from this function are in standard matrix format.

Speed can sometimes be increased by shifting from automatic sparse matrix determination to forced dense or sparse matrix projection. This will most likely occur when matrices have between 30 and 300 rows and columns. Defaults work best when matrices are very small and dense, or very large and sparse.

See Also

sensitivity3()

sensitivity3.lefkoMat()

sensitivity3.dgCMatrix()

sensitivity3.list()

sensitivity3.lefkoMatList()

Examples

```
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
matvector <- c(0, 0, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)

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

lathvert <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "Volume88", repstracol = "FCODE88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframe, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, censor = TRUE)

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "Sd", "Sdl", "mat"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "rep", "rep", "Sdl"),
  stage1 = c("Sd", "rep", "Sd", "rep", "npr", "npr", "Sd"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, "mat"),
```

```
  eststage2 = c(NA, NA, NA, NA, NA, NA, "Sdl"),
  eststage1 = c(NA, NA, NA, NA, NA, NA, "NotAlive"),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, 0.345, 0.054, NA),
  type = c(1, 1, 1, 1, 3, 3, 1), type_t12 = c(1, 2, 1, 2, 1, 1, 1),
  stageframe = lathframe, historical = TRUE)

ehrlen3 <- rlefko3(data = lathvert, stageframe = lathframe, year = "all",
  stages = c("stage3", "stage2", "stage1"), supplement = lathsupp3,
  yearcol = "year2", indivcol = "individ")

ehrlen3mean <- lmean(ehrlen3)
sensitivity3(ehrlen3mean$A[[1]])
```

`sf_create` *Create Stageframe for Population Matrix Projection Analysis*

Description

Function `sf_create()` returns a data frame describing each ahistorical life history stage in the life history model. This data frame can be used as input into MPM creation functions including `flefko3()`, `flefko2()`, `aflefko2()`, `rlefko3()`, `rlefko2()`, and `arlefko2()`, in which it determines how each stage is treated during matrix estimation.

Usage

```
sf_create(
  sizes,
  stagenames = NULL,
  sizesb = NULL,
  sizesc = NULL,
  repstatus = NULL,
  obsstatus = NULL,
  propstatus = NULL,
  matstatus = NULL,
  immstatus = NULL,
  minage = NULL,
  maxage = NULL,
  indataset = NULL,
  sizemin = NULL,
  sizebmin = NULL,
  sizecmin = NULL,
  sizemax = NULL,
  sizebmax = NULL,
  sizecmax = NULL,
  binhalfwidth = NULL,
  binhalfwidthb = NULL,
```

```
  binhalfwidthc = NULL,
  group = NULL,
  comments = NULL,
  roundsize = 5L,
  roundsizeb = 5L,
  roundsizec = 5L,
  ipmbins = 100L,
  ipmbinsb = NA_integer_,
  ipmbinsc = NA_integer_
)
```

Arguments

sizes	A numeric vector of the typical or representative size of each life history stage. If making function-based MPMs, then this may be a vector composed of the midpoints of each size bin, or simply of sizes characteristic of the size bins. If denoting the boundary of an automated size classification group, then should denote the absolute minimum size of that group, or the absolute size of that group (see Notes).
stagenames	A vector of stage names, in the same order as elements in sizes. Can also be set to ipm for automated size classification (see Notes section).
sizesb	An optional numeric vector for a second size metric for each life history stage. Only to be used if stages are defined by at least two size metrics in all cases. Same issues apply as in sizes.
sizesc	An optional numeric vector for a third size metric for each life history stage. Only to be used if stages are defined by at least three size metrics in all cases. Same issues apply as in sizes.
repstatus	A vector denoting the binomial reproductive status of each life history stage. Defaults to 1.
obsstatus	A vector denoting the binomial observation status of each life history stage. Defaults to 1, but may be changed for unobservable stages.
propstatus	A vector denoting whether each life history stage is a propagule. Such stages are generally only used in fecundity estimation. Defaults to 0.
matstatus	A vector denoting whether each stage is mature. Must be composed of binomial values if given. Defaults to 1 for all stages defined in sizes.
immstatus	A vector denoting whether each stage is immature. Must be composed of binomial values if given. Defaults to the complement of vector matstatus.
minage	An optional vector denoting the minimum age at which a stage can occur. Only used in age x stage matrix development. Defaults to NA.
maxage	An optional vector denoting the maximum age at which a stage should occur. Only used in age x stage matrix development. Defaults to NA.
indataset	A vector designating which stages are found within the dataset. While rlefko2() and rlefko3() can use all stages in the input dataset, flefko3() and flefko2() can only handle size-classified stages with non-overlapping combinations of size and status variables. Stages that do not actually exist within the dataset should be marked as 0 in this vector.

sizemin	A vector giving the absolute minimum values corresponding to each size in the sizes vector. Requires associated values for sizemax if used. Only required if not using binhalfwidth.
sizebmin	A vector giving the absolute minimum values corresponding to each size in the sizesb vector. Requires associated values for sizebmax if used. Only required if not using binhalfwidthb.
sizecmin	A vector giving the absolute minimum values corresponding to each size in the sizesc vector. Requires associated values for sizecmax if used. Only required if not using binhalfwidthc.
sizemax	A vector giving the absolute maximum values corresponding to each size in the sizes vector. Requires associated values for sizemin if used. Only required if not using binhalfwidth.
sizebmax	A vector giving the absolute maximum values corresponding to each size in the sizesb vector. Requires associated values for sizebmin if used. Only required if not using binhalfwidthb.
sizecmax	A vector giving the absolute maximum values corresponding to each size in the sizesc vector. Requires associated values for sizecmin if used. Only required if not using binhalfwidthc.
binhalfwidth	A numeric vector giving the half-width of size bins. Required if sizemin and sizemax are not used. Defaults to 0.5 for all sizes.
binhalfwidthb	A numeric vector giving the half-width of size bins used for the optional second size metric. Required if sizebmin and sizebmax are not used but two or three size classes are used. Defaults to 0.5 for all sizes.
binhalfwidthc	A numeric vector giving the half-width of size bins used for the optional third size metric. Required if sizecmin and sizecmax are not used but three size classes are used. Defaults to 0.5 for all sizes.
group	An integer vector providing information on each respective stage's size classification group. If used, then function-based MPM creation functions flefko2(), flefko3(), and aflefko2() will estimate transitions only within these groups and for allowed cross-group transitions noted within the supplement table. Defaults to 0.
comments	An optional vector of text entries holding useful text descriptions of all stages.
roundsize	This parameter sets the precision of size classification, and equals the number of digits used in rounding sizes. Defaults to 5.
roundsizeb	This parameter sets the precision of size classification in the optional second size metric, and equals the number of digits used in rounding sizes. Defaults to 5.
roundsizec	This parameter sets the precision of size classification in the optional third size metric, and equals the number of digits used in rounding sizes. Defaults to 5.
ipmbins	An integer giving the number of size bins to create using the primary size classification variable. This number is in addition to any stages that are not size classified. Defaults to 100, and numbers greater than this yield a warning about the loss of statistical power and increasing chance of matrix over-parameterization resulting from increasing numbers of stages.

ipmbinsb — An optional integer giving the number of size bins to create using the secondary size classification variable. This number is in addition to any stages that are not size classified, as well as in addition to any automated size classification using the primary and tertiary size variables. Defaults to NA, and must be set to a positive integer for automated size classification to progress.

ipmbinsc — An optional integer giving the number of size bins to create using the tertiary size classification variable. This number is in addition to any stages that are not size classified, as well as in addition to any automated size classification using the primary and secondary size variables. Defaults to NA, and must be set to a positive integer for automated size classification to progress.

Value

A data frame of class stageframe, which includes information on the stage name, size, reproductive status, observation status, propagule status, immaturity status, maturity status, presence within the core dataset, stage group classification, raw bin half-width, and the minimum, center, and maximum of each size bin, as well as its width. If minimum and maximum ages were specified, then these are also included. Also includes an empty string variable that can be used to describe stages meaningfully. This object can be used as the stageframe input for flefko3() flefko2(), rlefko3(), and rlefko2().

Variables in this data frame include the following:

stage — The unique names of the stages to be analyzed.

size — The typical or representative size at which each stage occurs.

size_b — Size at which each stage occurs in terms of a second size variable, if one exists.

size_c — Size at which each stage occurs in terms of a third size variable, if one exists.

min_age — The minimum age at which the stage may occur.

max_age — The maximum age at which the stage may occur.

repstatus — A binomial variable showing whether each stage is reproductive.

obsstatus — A binomial variable showing whether each stage is observable.

propstatus — A binomial variable showing whether each stage is a propagule.

immstatus — A binomial variable showing whether each stage can occur as immature.

matstatus — A binomial variable showing whether each stage occurs in maturity.

indataset — A binomial variable describing whether each stage occurs in the input dataset.

binhalfwidth_raw — The half-width of the size bin, as input.

sizebin_min — The minimum size at which the stage may occur.

sizebin_max — The maximum size at which the stage may occur.

sizebin_center — The midpoint of the size bin at which the stage may occur.

sizebin_width — The width of the size bin corresponding to the stage.

binhalfwidthb_raw — The half-width of the size bin of a second size variable, as input.

sizebinb_min — The minimum size at which the stage may occur.

sizebinb_max	The maximum size at which the stage may occur.
sizebinb_center	The midpoint of the size bin at which the stage may occur, in terms of a second size variable.
sizebinb_width	The width of the size bin corresponding to the stage, in terms of a second size variable.
binhalfwidthc_raw	The half-width of the size bin of a third size variable, as input.
sizebinc_min	The minimum size at which the stage may occur, in terms of a third size variable.
sizebinc_max	The maximum size at which the stage may occur, in terms of a third size variable.
sizebinc_center	The midpoint of the size bin at which the stage may occur, in terms of a third size variable.
sizebinc_width	The width of the size bin corresponding to the stage, in terms of a third size variable.
group	An integer denoting the size classification group that the stage falls within.
comments	A text field for stage descriptions.

Notes

Vectors used to create a stageframe may not mix NA values with non-NA values.

If an IPM or function-based matrix with automated size classification is desired, then two stages that occur within the dataset and represent the lower and upper size limits of the IPM must be marked with ipm in the stagenames vector. These stages should have all characteristics other than size equal, and the size input for whichever size will be classified automatically must include the minimum in one stage and the maximum in the other. The actual characteristics of the first stage encountered in the inputs will be used as the template for the creation of these sizes. Note that ipm refers to size classification with the primary size variable. To automate size classification with the secondary size variable, use ipmb, and to automate size classification with the tertiary size variable, use ipmc. To nest automated size classifications, use ipmab for the primary and secondary size variables, ipmac for the primary and tertiary size variables, ipmbc for the secondary and tertiary size variables, and ipmabc for all three size variables. The primary size variable can also be set with ipma.

If two or more groups of stages, each with its own characteristics, are to be developed for an IPM or function-based MPM, then an even number of stages with two stages marking the minimum and maximum size of each group should be marked with the same code as given above, with all other characteristics equal within each group.

Stage classification groups set with the group variable create zones within function-based matrices in which survival transitions are estimated. These groups should not be set if transitions are possible between all stages regardless of group. To denote specific transitions as estimable between stage groups, use the supplemental() function.

If importing an IPM rather than building one with lefko3: Using the vrm_import approach to building function-based MPMs with provided linear model slope coefficients requires careful attention to the stageframe. Although no hfv data frame needs to be entered in this instance, stages for which vital rates are to be estimated via linear models parameterized with coefficients provided via function vrm_import() should be marked as occurring within the dataset. Stages for which the provided coefficients should not be used should be marked as not occurring within the dataset.

Examples

```
# Lathyrus example
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
matvector <- c(0, 0, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)

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

# Cypripedium example
data(cypdata)

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)
```

`sf_distrib` *Test Overdispersion and Zero Inflation in Size and Fecundity Distributions*

Description

Function `sf_distrib` takes a historically formatted vertical data as input and tests whether size and fecundity data are dispersed according to a Poisson distribution (where mean = variance), and whether the number of 0s exceeds expectations. This function is now deprecated in favor of function `hfv_qc()`.

Usage

```
sf_distrib(
  data,
  sizea = NA,
  sizeb = NA,
  sizec = NA,
  obs3 = NA,
  fec = NA,
  repst = NA,
  zisizea = TRUE,
  zisizeb = TRUE,
  zisizec = TRUE,
  zifec = TRUE,
  fectime = 2,
  show.size = TRUE,
  show.fec = TRUE
)
```

Arguments

data	A historical vertical data file, which is a data frame of class hfvdata.
sizea	A vector holding the name or column number of the variables corresponding to primary size in occasions *t*+1 and *t*. Input only if sizea is to be tested.
sizeb	A vector holding the name or column number of the variables corresponding to secondary size in occasions *t*+1 and *t*. Input only if sizeb is to be tested.
sizec	A vector holding the name or column number of the variables corresponding to tertiary size in occasions *t*+1 and *t*. Input only if sizec is to be tested.
obs3	The name or column number of the variable corresponding to observation status in occasion *t*+1. This should be used if observation status will be used as a vital rate to absorb states of size = 0.
fec	A vector holding the names or column numbers of the variables corresponding to in occasions *t*+1 and *t*. Input only if fec is to be tested.
repst	A vector holding the names or column numbers of the variables corresponding to reproductive status in occasions *t*+1 and *t*. If not provided, then fecundity will be tested without subsetting to only reproductive individuals.
zisizea	A logical value indicating whether to conduct a test of zero inflation in primary size. Defaults to TRUE.
zisizeb	A logical value indicating whether to conduct a test of zero inflation in secondary size. Defaults to TRUE.
zisizec	A logical value indicating whether to conduct a test of zero inflation in tertiary size. Defaults to TRUE.
zifec	A logical value indicating whether to conduct a test of zero inflation in fecundity. Defaults to TRUE.
fectime	An integer indicating whether to treat fecundity as occurring in time *t* (2) or time *t*+1 (3). Defaults to 2.

show.size A logical value indicating whether to show the output for tests of size. Defaults to TRUE.

show.fec A logical value indicating whether to show the output for tests of fecundity. Defaults to TRUE.

Value

Produces text describing the degree and significance of difference from expected dispersion, and the degree and significance of zero inflation. The tests are chi-squared score tests based on the expectations of mean = variance, and 0s as abundant as predicted by the value of lambda estimated from the dataset. See van der Broek (1995) for more details.

Notes

This function subsets the data in the same way as modelsearch() before testing underlying distributions, making the output much more appropriate than a simple analysis of size and fecundity variables in data.

The specific test used for overdispersion is a chi-squared test of the dispersion parameter estimated using a generalized linear model predicting the response given size in occasion *t*, under a quasi-Poisson distribution.

The specific test used for zero-inflation is the chi-squared test presented in van der Broek (1995).

Examples

```
data(lathyrus)

sizevector <- c(0, 4.6, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 1, 2, 3, 4, 5, 6, 7, 8,
  9)
stagevector <- c("Sd", "Sdl", "Dorm", "Sz1nr", "Sz2nr", "Sz3nr", "Sz4nr",
  "Sz5nr", "Sz6nr", "Sz7nr", "Sz8nr", "Sz9nr", "Sz1r", "Sz2r", "Sz3r",
  "Sz4r", "Sz5r", "Sz6r", "Sz7r", "Sz8r", "Sz9r")
repvector <- c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1)
obsvector <- c(0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
matvector <- c(0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0)
indataset <- c(0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 4.6, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
  0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5)

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

lathvertln <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "lnVol88", repstracol = "Intactseed88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
```

```
  nonobsacol = "Dormant1988", stageassign = lathframeln, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, NAas0 = TRUE, censor = TRUE)

lathvertln$feca2 <- round(lathvertln$feca2)
lathvertln$feca1 <- round(lathvertln$feca1)
lathvertln$feca3 <- round(lathvertln$feca3)

sf_distrib(lathvertln, sizea = c("sizea3", "sizea2"), fec = c("feca3", "feca2"),
  repst = c("repstatus3", "repstatus2"), zifec = FALSE)
```

---

sf_skeleton *Create Skeleton Stageframe*

Description

Function `sf_skeleton()` creates a skeleton `stageframe` object.

Usage

```
sf_skeleton(stages, standard = TRUE)
```

Arguments

`stages`	The number of stages, as an integer.
`standard`	A logical value indicating whether to create a standard `stageframe` object (`TRUE`, the default), or a reassessed `stageframe` object as created by function `mpm_create()` (`FALSE`).

Value

A data frame of class `stageframe`.

---

slambda3 *Estimate Stochastic Population Growth Rate*

Description

Function `slambda3()` estimates the stochastic population growth rate, a, defined as the long-term arithmetic mean of the log population growth rate estimated per simulated occasion. This function can handle lefkoMat objects, lefkoMatList objects, and lists of full A matrices as input.

Usage

```
slambda3(
  mpm,
  times = 10000L,
  historical = FALSE,
  tweights = NULL,
  force_sparse = NULL
)
```

Arguments

mpm	A matrix projection model of class lefkoMat, a bootstrapped MPM object of class lefkoMatList, or a simple list of full matrix projection matrices.
times	Number of occasions to iterate. Defaults to 10000.
historical	An optional logical value only used if object mpm is a list of matrices, rather than a lefkoMat object. Defaults to FALSE for the former case, and overridden by information supplied in the lefkoMat object for the latter case.
tweights	An optional numeric vector or matrix denoting the probabilities of choosing each matrix in a stochastic projection. If a matrix is input, 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 a vector is input, then the choice of annual matrix is assumed to be independent of the current matrix. Defaults to equal weighting among matrices.
force_sparse	A text string indicating whether to force sparse matrix encoding ("yes") or not ("no") if the MPM is composed of simple matrices. Defaults to "auto", in which case sparse matrix encoding is used with simple square matrices with at least 50 rows and no more than 50% of elements with values greater than zero.

Value

A data frame with the following variables:

replicate	The bootstrapped replicate. Only provided if a lefkoMatList object is entered in argument mpm.
pop	The identity of the population.
patch	The identity of the patch.
a	Estimate of stochastic growth rate, estimated as the arithmetic mean of the log population growth rate across simulated occasions.
var	The estimated variance of a.
sd	The standard deviation of a.
se	The standard error of a.

Notes

The log stochastic population growth rate, a, is as given in equation 2 of Tuljapurkar, Horvitz, and Pascarella 2003. This term is estimated via projection of randomly sampled matrices, similarly to the procedure outlined in Box 7.4 of Morris and Doak (2002).

Stochastic growth rate is estimated both at the patch level and at the population level. Population level estimates will be noted at the end of the data frame with 0 entries for patch designation.

Weightings given in `tweights` do not need to sum to 1. Final weightings used will be based on the proportion per element of the sum of elements in the user-supplied vector.

Speed can sometimes be increased by shifting from automatic sparse matrix determination to forced dense or sparse matrix projection. This will most likely occur when matrices have between 30 and 300 rows and columns. Defaults work best when matrices are very small and dense, or very large and sparse.

Examples

```
data(cypdata)

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 = 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)

cypraw_boot <- bootstrap3(cypraw_v1, reps = 3)

cypsupp3r <- supplemental(stage3 = c("SD", "SD", "P1", "P1", "P2", "P3", "SL",
    "D", "XSm", "Sm", "D", "XSm", "Sm", "mat", "mat", "mat", "SD", "P1"),
  stage2 = c("SD", "SD", "SD", "SD", "P1", "P2", "P3", "SL", "SL", "SL", "SL",
    "SL", "SL", "D", "XSm", "Sm", "rep", "rep"),
  stage1 = c("SD", "rep", "SD", "rep", "SD", "P1", "P2", "P3", "P3", "P3",
    "SL", "SL", "SL", "SL", "SL", "SL", "mat", "mat"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, NA, "D", "XSm", "Sm", "D", "XSm", "Sm",
    "mat", "mat", "mat", NA, NA),
```

```
  eststage2 = c(NA, NA, NA, NA, NA, NA, NA, "XSm", "XSm", "XSm", "XSm", "XSm",
    "XSm", "D", "XSm", "Sm", NA, NA),
  eststage1 = c(NA, NA, NA, NA, NA, NA, NA, "XSm", "XSm", "XSm", "XSm", "XSm",
    "XSm", "XSm", "XSm", "XSm", NA, NA),
  givenrate = c(0.1, 0.1, 0.2, 0.2, 0.2, 0.2, 0.25, NA, NA, NA, NA, NA, NA,
    NA, NA, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
    NA, 0.5, 0.5),
  type = c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3),
  type_t12 = c(1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1),
  stageframe = cypframe_raw, historical = TRUE)

cypmatrix3r <- rlefko3(data = cypraw_v1, stageframe = cypframe_raw,
  year = "all", patch = "all", stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added", "size1added"),
  supplement = cypsupp3r, yearcol = "year2",
  patchcol = "patchid", indivcol = "individ")

cypstoch <- slambda3(cypmatrix3r)

cypmatrix3r_boot <- rlefko3(data = cypraw_boot, stageframe = cypframe_raw,
year = "all", patch = "all", stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added", "size1added"),
  supplement = cypsupp3r, yearcol = "year2",
  patchcol = "patchid", indivcol = "individ")

cypstoch_boot <- slambda3(cypmatrix3r_boot)
```

`stablestage3` *Estimate Stable Stage Distribution*

Description

`stablestage3()` is a generic function that returns the stable stage distribution for a population projection matrix or set of matrices. This function is made to handle very large and sparse matrices supplied as `lefkoMat` objects or as individual matrices, and can be used with large historical matrices, IPMs, age x stage matrices, as well as ahistorical matrices.

Usage

```
stablestage3(mats, ...)
```

Arguments

`mats`	A lefkoMat object, a population projection matrix, or a list of population projection matrices for which the stable stage distribution is desired.
`...`	Other parameters.

Value

The value returned depends on the class of the `mats` argument. See related functions for details.

See Also

`stablestage3.lefkoMat()`

`stablestage3.list()`

`stablestage3.matrix()`

`stablestage3.dgCMatrix()`

Examples

```
# Lathyrus deterministic example
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
matvector <- c(0, 0, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)

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

lathvert <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "Volume88", repstracol = "FCODE88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframe, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, censor = TRUE)

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "Sd", "Sdl", "mat"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "rep", "rep", "Sdl"),
  stage1 = c("Sd", "rep", "Sd", "rep", "npr", "npr", "Sd"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, "mat"),
  eststage2 = c(NA, NA, NA, NA, NA, NA, "Sdl"),
  eststage1 = c(NA, NA, NA, NA, NA, NA, "NotAlive"),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, 0.345, 0.054, NA),
  type = c(1, 1, 1, 1, 3, 3, 1), type_t12 = c(1, 2, 1, 2, 1, 1, 1),
  stageframe = lathframe, historical = TRUE)

ehrlen3 <- rlefko3(data = lathvert, stageframe = lathframe, year = "all",
  stages = c("stage3", "stage2", "stage1"), supplement = lathsupp3,
  yearcol = "year2", indivcol = "individ")
```

```
ehrlen3mean <- lmean(ehrlen3)
stablestage3(ehrlen3mean)

# Cypripedium stochastic example
data(cypdata)

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 = 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)

# Here we use supplemental() to provide overwrite and reproductive info
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", patch = "all", stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added"), supplement = cypsupp2r,
  yearcol = "year2", patchcol = "patchid", indivcol = "individ")

stablestage3(cypmatrix2r, stochastic = TRUE)
```

`stablestage3.dgCMatrix` *Estimate Stable Stage Distribution of a Single Population Projection Matrix*

Description

`stablestage3.dgCMatrix()` returns the stable stage distribution for a sparse population projection matrix.

Usage

```
## S3 method for class 'dgCMatrix'
stablestage3(mats, ...)
```

Arguments

`mats`	A population projection matrix of class `dgCMatrix`.
`...`	Other parameters.

Value

This function returns the stable stage distribution corresponding to the input matrix.

See Also

`stablestage3()`

`stablestage3.lefkoMat()`

`stablestage3.list()`

`stablestage3.matrix()`

`stablestage3.lefkoMatList()`

Examples

```
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
matvector <- c(0, 0, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)

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

```
  propstatus = propvector)

lathvert <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "Volume88", repstracol = "FCODE88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframe, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, censor = TRUE)

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "Sd", "Sdl", "mat"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "rep", "rep", "Sdl"),
  stage1 = c("Sd", "rep", "Sd", "rep", "npr", "npr", "Sd"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, "mat"),
  eststage2 = c(NA, NA, NA, NA, NA, NA, "Sdl"),
  eststage1 = c(NA, NA, NA, NA, NA, NA, "NotAlive"),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, 0.345, 0.054, NA),
  type = c(1, 1, 1, 1, 3, 3, 1), type_t12 = c(1, 2, 1, 2, 1, 1, 1),
  stageframe = lathframe, historical = TRUE)

ehrlen3 <- rlefko3(data = lathvert, stageframe = lathframe, year = "all",
  stages = c("stage3", "stage2", "stage1"), supplement = lathsupp3,
  yearcol = "year2", indivcol = "individ", sparse_output = TRUE)

stablestage3(ehrlen3$A[[1]])
```

`stablestage3.lefkoMat` *Estimate Stable Stage Distribution of Matrices in lefkoMat Object*

Description

`stablestage3.lefkoMat()` returns the deterministic stable stage distributions of all `A` matrices in an object of class `lefkoMat`, as well as the long-run projected mean stage distribution in stochastic analysis. This function can handle large and sparse matrices, and so can be used with large historical matrices, IPMs, age x stage matrices, as well as ahistorical matrices.

Usage

```
## S3 method for class 'lefkoMat'
stablestage3(
  mats,
  stochastic = FALSE,
  times = 10000,
  tweights = NA,
  seed = NA,
  force_sparse = "auto",
  ...
)
```

Arguments

mats	An object of class lefkoMat.
stochastic	A logical value indicating whether to use deterministic (FALSE) or stochastic (TRUE) analysis. Defaults to FALSE.
times	An integer variable indicating number of occasions to project if using stochastic analysis. Defaults to 10000.
tweights	An optional numeric vector or matrix denoting the probabilities of choosing each matrix in a stochastic projection. If a matrix is input, 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 a vector is input, then the choice of annual matrix is assumed to be independent of the current matrix. Defaults to equal weighting among matrices.
seed	A number to use as a random number seed in stochastic projection.
force_sparse	A text string indicating whether to use sparse matrix encoding ("yes") if standard matrices are provided. Defaults to "auto", in which case sparse matrix encoding is used with square matrices with at least 50 rows and no more than 50% of elements with values greater than zero.
...	Other parameters.

Value

This function returns the stable stage distributions (and long-run mean stage distributions in stochastic analysis) corresponding to the matrices in a lefkoMat object.

The output depends on whether the lefkoMat object used as input is ahistorical or historical, and whether the analysis is deterministic or stochastic. If deterministic and ahistorical, then a single data frame is output, which includes the number of the matrix within the A element of the input lefkoMat object, followed by the stage id (numeric and assigned through sf_create()), the stage name, and the estimated proportion of the stable stage distribution (ss_prop). If stochastic and ahistorical, then a single data frame is output starting with the number of the population-patch (matrix_set), a string concatenating the names of the population and the patch (poppatch), the assigned stage id number (stage_id), and the stage name (stage), and the long-run average stage distribution (ss_prop).

If a historical matrix is used as input, then two data frames are output into a list object. The hist element describes the historical stage-pair distribution, while the ahist element describes the stage distribution. If deterministic, then hist contains a data frame including the matrix number (matrix), the numeric stage designations for stages in occasions *t* and *t*-1, (stage_id_2 and stage_id_1, respectively), followed by the respective stage names (stage_2 and stage_1), and ending with the estimated stable stage-pair distribution. The associated ahist element is as before. If stochastic, then the hist element contains a single data frame with the number of the population-patch (matrix_set), a string concatenating the names of the population and the patch (poppatch), the assigned stage id numbers in times *t* and *t*-1 (stage_id_2 and stage_id_2, respectively), and the associated stage names (stage_2 and stage_1, respectively), and the long-run average stage distribution (ss_prop). The associated ahist element is as before in the ahistorical, stochastic case.

In addition to the data frames noted above, stochastic analysis will result in the additional output of a list of matrices containing the actual projected stage distributions across all projected occasions, in the order of population-patch combinations in the lefkoMat input.

Notes

In stochastic analysis, the projected mean distribution is the arithmetic mean across the final 1000 projected occasions if the simulation is at least 2000 projected occasions long. If between 500 and 2000 projected occasions long, then only the final 200 are used, and if fewer than 500 occasions are used, then all are used. Note that because stage distributions in stochastic simulations can change greatly in the initial portion of the run, we encourage a minimum of 2000 projected occasions per simulation, with 10000 preferred.

Speed can sometimes be increased by shifting from automatic sparse matrix determination to forced dense or sparse matrix projection. This will most likely occur when matrices have between 30 and 300 rows and columns. Defaults work best when matrices are very small and dense, or very large and sparse.

See Also

stablestage3()

stablestage3.list()

stablestage3.matrix()

stablestage3.dgCMatrix()

stablestage3.lefkoMatList()

Examples

```
# Lathyrus deterministic example
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
matvector <- c(0, 0, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)

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

lathvert <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "Volume88", repstracol = "FCODE88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframe, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, censor = TRUE)

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "Sd", "Sdl", "mat"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "rep", "rep", "Sdl"),
```

```
  stage1 = c("Sd", "rep", "Sd", "rep", "npr", "npr", "Sd"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, "mat"),
  eststage2 = c(NA, NA, NA, NA, NA, NA, "Sdl"),
  eststage1 = c(NA, NA, NA, NA, NA, NA, "NotAlive"),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, 0.345, 0.054, NA),
  type = c(1, 1, 1, 1, 3, 3, 1), type_t12 = c(1, 2, 1, 2, 1, 1, 1),
  stageframe = lathframe, historical = TRUE)

ehrlen3 <- rlefko3(data = lathvert, stageframe = lathframe, year = "all",
  stages = c("stage3", "stage2", "stage1"), supplement = lathsupp3,
  yearcol = "year2", indivcol = "individ")

ehrlen3mean <- lmean(ehrlen3)
stablestage3(ehrlen3mean)

# Cypripedium stochastic example
data(cypdata)

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 = 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)

# Here we use supplemental() to provide overwrite and reproductive info
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", patch = "all", stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added"), supplement = cypsupp2r,
  yearcol = "year2", patchcol = "patchid", indivcol = "individ")

stablestage3(cypmatrix2r, stochastic = TRUE)
```

`stablestage3.lefkoMatList` *Estimate Stable Stage Distribution of Matrices in lefkoMatList Object*

Description

`stablestage3.lefkoMatList()` returns the deterministic stable stage distributions of all `A` matrices in all objects of class `lefkoMat` stored within the entered `lefkoMatList` object, as well as the long-run projected mean stage distribution in stochastic analysis. This function can handle large and sparse matrices, and so can be used with large historical matrices, IPMs, age x stage matrices, as well as ahistorical matrices.

Usage

```
## S3 method for class 'lefkoMatList'
stablestage3(
  mats,
  stochastic = FALSE,
  times = 10000,
  tweights = NA,
  seed = NA,
  force_sparse = "auto",
  ...
)
```

Arguments

`mats`	An object of class `lefkoMatList`.
`stochastic`	A logical value indicating whether to use deterministic (`FALSE`) or stochastic (`TRUE`) analysis. Defaults to `FALSE`.
`times`	An integer variable indicating number of occasions to project if using stochastic analysis. Defaults to 10000.
`tweights`	An optional numeric vector or matrix denoting the probabilities of choosing each matrix in a stochastic projection. If a matrix is input, 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 a vector is input, then the choice of annual matrix is assumed to be independent of the current matrix. Defaults to equal weighting among matrices.

seed	A number to use as a random number seed in stochastic projection.
force_sparse	A text string indicating whether to use sparse matrix encoding ("yes") if standard matrices are provided. Defaults to "auto", in which case sparse matrix encoding is used with square matrices with at least 50 rows and no more than 50% of elements with values greater than zero.
...	Other parameters.

Value

This function returns a list with two elements. The first is the mean stable stage distribution (and long-run mean stage distributions from stochastic analysis), and the second is a list of stable stage distributions (and long-run mean stage distributions from stochastic analysis) corresponding to the lefkoMat objects in the original mats list.

The format of distributions in both cases depends on whether the lefkoMat objects used as input are ahistorical or historical, and whether the analysis is deterministic or stochastic. If deterministic and ahistorical, then a single data frame is output, which includes the number of the matrix within the A element of the input lefkoMat object, followed by the stage id (numeric and assigned through sf_create()), the stage name, and the estimated proportion of the stable stage distribution (ss_prop). If stochastic and ahistorical, then a single data frame is output starting with the number of the population-patch (matrix_set), a string concatenating the names of the population and the patch (poppatch), the assigned stage id number (stage_id), and the stage name (stage), and the long-run average stage distribution (ss_prop).

If historical matrices are used as input, then two data frames are output into a list object. The hist element describes the historical stage-pair distribution, while the ahist element describes the stage distribution. If deterministic, then hist contains a data frame including the matrix number (matrix), the numeric stage designations for stages in occasions *t* and *t*-1, (stage_id_2 and stage_id_1, respectively), followed by the respective stage names (stage_2 and stage_1), and ending with the estimated stable stage-pair distribution. The associated ahist element is as before. If stochastic, then the hist element contains a single data frame with the number of the population-patch (matrix_set), a string concatenating the names of the population and the patch (poppatch), the assigned stage id numbers in times *t* and *t*-1 (stage_id_2 and stage_id_2, respectively), and the associated stage names (stage_2 and stage_1, respectively), and the long-run average stage distribution (ss_prop). The associated ahist element is as before in the ahistorical, stochastic case.

In addition to the data frames noted above, stochastic analysis will result in the additional output of a list of matrices containing the actual projected stage distributions across all projected occasions, in the order of population-patch combinations in the lefkoMat input.

Notes

In stochastic analysis, the projected mean distribution is the arithmetic mean across the final 1000 projected occasions if the simulation is at least 2000 projected occasions long. If between 500 and 2000 projected occasions long, then only the final 200 are used, and if fewer than 500 occasions are used, then all are used. Note that because stage distributions in stochastic simulations can change greatly in the initial portion of the run, we encourage a minimum of 2000 projected occasions per simulation, with 10000 preferred.

Speed can sometimes be increased by shifting from automatic sparse matrix determination to forced dense or sparse matrix projection. This will most likely occur when matrices have between 30 and

300 rows and columns. Defaults work best when matrices are very small and dense, or very large and sparse.

See Also

stablestage3()

stablestage3.list()

stablestage3.matrix()

stablestage3.dgCMatrix()

stablestage3.lefkoMat()

Examples

```
# Lathyrus deterministic example
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
matvector <- c(0, 0, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)

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

lathvert <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "Volume88", repstracol = "FCODE88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframe, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, censor = TRUE)

lathvert_boot <- bootstrap3(lathvert, reps = 3)

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "Sd", "Sdl", "mat"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "rep", "rep", "Sdl"),
  stage1 = c("Sd", "rep", "Sd", "rep", "npr", "npr", "Sd"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, "mat"),
  eststage2 = c(NA, NA, NA, NA, NA, NA, "Sdl"),
  eststage1 = c(NA, NA, NA, NA, NA, NA, "NotAlive"),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, 0.345, 0.054, NA),
  type = c(1, 1, 1, 1, 3, 3, 1), type_t12 = c(1, 2, 1, 2, 1, 1, 1),
  stageframe = lathframe, historical = TRUE)
```

```
ehrlen3_boot <- rlefko3(data = lathvert_boot, stageframe = lathframe,
  year = "all", stages = c("stage3", "stage2", "stage1"),
  supplement = lathsupp3, yearcol = "year2", indivcol = "individ")

ehrlen3mean <- lmean(ehrlen3_boot)
stablestage3(ehrlen3mean)

# Cypripedium stochastic example
data(cypdata)

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 = 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)

cypraw_v1_boot <- bootstrap3(cypraw_v1, reps = 3)

# Here we use supplemental() to provide overwrite and reproductive info
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_boot <- rlefko2(data = cypraw_v1_boot, stageframe = cypframe_raw,
  year = "all", patch = "all", stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added"), supplement = cypsupp2r,
  yearcol = "year2", patchcol = "patchid", indivcol = "individ")

stablestage3(cypmatrix2r_boot, stochastic = TRUE)
```

stablestage3.list *Estimate Stable Stage Distribution of a List of Projection Matrices*

Description

stablestage3.list() returns the stable stage distributions for stages in population projection matrices arranged in a general list. The function makes no assumptions about whether the matrix is ahistorical and simply provides stable stage distribution values corresponding to each row, meaning that the overall stable stage distribution of basic life history stages in a historical matrix are not provided (the stablestage3.lefkoMat() historical estimates these on the basis of stage description information provided in the lefkoMat object used as input in that function). This provided in the handle large and sparse matrices, and so can be used with large historical matrices, IPMs, age x stage matrices, as well as smaller ahistorical matrices.

Usage

```
## S3 method for class 'list'
stablestage3(
  mats,
  stochastic = FALSE,
  times = 10000,
  tweights = NA,
  seed = NA,
  force_sparse = "auto",
  ...
)
```

Arguments

mats	A list of population projection matrices, all in either class matrix or class dgCMatrix.
stochastic	A logical value indicating whether to use deterministic (FALSE) or stochastic (TRUE) analysis. Defaults to FALSE.
times	An integer variable indicating number of occasions to project if using stochastic analysis. Defaults to 10000.
tweights	An optional numeric vector or matrix denoting the probabilities of choosing each matrix in a stochastic projection. If a matrix is input, 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 a vector is input, then the choice of annual matrix is assumed to be independent of the current matrix. Defaults to equal weighting among matrices.
seed	A number to use as a random number seed in stochastic projection.

force_sparse — A text string indicating whether to use sparse matrix encoding ("yes") when supplied with standard matrices. Defaults to "auto", in which case sparse matrix encoding is used with square matrices with at least 50 rows and no more than 50% of elements with values greater than zero.

... — Other parameters.

Value

This function returns a list of vector data frames characterizing the stable stage distributions for stages of each population projection matrix.

Notes

Speed can sometimes be increased by shifting from automatic sparse matrix determination to forced dense or sparse matrix projection. This will most likely occur when matrices have between 30 and 300 rows and columns. Defaults work best when matrices are very small and dense, or very large and sparse.

See Also

stablestage3()

stablestage3.lefkoMat()

stablestage3.matrix()

stablestage3.dgCMatrix()

Examples

```
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
matvector <- c(0, 0, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)

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

lathvert <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "Volume88", repstracol = "FCODE88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframe, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, censor = TRUE)
```

```
lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "Sd", "Sdl", "mat"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "rep", "rep", "Sdl"),
  stage1 = c("Sd", "rep", "Sd", "rep", "npr", "npr", "Sd"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, "mat"),
  eststage2 = c(NA, NA, NA, NA, NA, NA, "Sdl"),
  eststage1 = c(NA, NA, NA, NA, NA, NA, "NotAlive"),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, 0.345, 0.054, NA),
  type = c(1, 1, 1, 1, 3, 3, 1), type_t12 = c(1, 2, 1, 2, 1, 1, 1),
  stageframe = lathframe, historical = TRUE)

ehrlen3 <- rlefko3(data = lathvert, stageframe = lathframe, year = "all",
  stages = c("stage3", "stage2", "stage1"), supplement = lathsupp3,
  yearcol = "year2", indivcol = "individ")

ehrlen3mean <- lmean(ehrlen3)
stablestage3(ehrlen3mean$A)
```

`stablestage3.matrix` *Estimate Stable Stage Distribution of a Single Population Projection Matrix*

Description

`stablestage3.matrix()` returns the stable stage distribution for a population projection matrix. This function can handle large and sparse matrices, and so can be used with large historical matrices, IPMs, age x stage matrices, as well as smaller ahistorical matrices.

Usage

```
## S3 method for class 'matrix'
stablestage3(mats, force_sparse = "auto", ...)
```

Arguments

`mats`	A population projection matrix of class `matrix`.
`force_sparse`	A text string indicating whether to use sparse matrix encoding (`"yes"`) when supplied with standard matrices. Defaults to `"auto"`, in which case sparse matrix encoding is used with square matrices with at least 50 rows and no more than 50% of elements with values greater than zero.
`...`	Other parameters.

Value

This function returns the stable stage distribution corresponding to the input matrix.

Notes

Speed can sometimes be increased by shifting from automatic sparse matrix determination to forced dense or sparse matrix projection. This will most likely occur when matrices have between 30 and 300 rows and columns. Defaults work best when matrices are very small and dense, or very large and sparse.

See Also

stablestage3()

stablestage3.lefkoMat()

stablestage3.lefkoMatList()

stablestage3.list()

stablestage3.dgCMatrix()

Examples

```
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
matvector <- c(0, 0, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)

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

lathvert <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "Volume88", repstracol = "FCODE88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframe, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, censor = TRUE)

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "Sd", "Sdl", "mat"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "rep", "rep", "Sdl"),
  stage1 = c("Sd", "rep", "Sd", "rep", "npr", "npr", "Sd"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, "mat"),
  eststage2 = c(NA, NA, NA, NA, NA, NA, "Sdl"),
  eststage1 = c(NA, NA, NA, NA, NA, NA, "NotAlive"),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, 0.345, 0.054, NA),
  type = c(1, 1, 1, 1, 3, 3, 1), type_t12 = c(1, 2, 1, 2, 1, 1, 1),
  stageframe = lathframe, historical = TRUE)
```

```
ehrlen3 <- rlefko3(data = lathvert, stageframe = lathframe, year = "all",
  stages = c("stage3", "stage2", "stage1"), supplement = lathsupp3,
  yearcol = "year2", indivcol = "individ")

ehrlen3mean <- lmean(ehrlen3)
stablestage3(ehrlen3mean$A[[1]])
```

---

stage_weight *Create a Vector of Stage Weights for Density Dependence Calculations*

Description

Function stage_weight() creates a data frame summarizing the weighting of individuals of different stages, stage-stage combinations, ages, or age-stages relative to one another in terms of the total population size. Stage weights can be thought of as Lotka-Volterra coefficients applied to individuals of different stages within a single population.

Usage

```
stage_weight(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 lefkoEq, with four objects, which can be used as input in function projection3() and f_projection3(). 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

Users should generally consider which stage to set as the reference, which should be designated with a value of 1.

In some life histories, certain stages may not count against population size with regards to the kind and value of density dependence decided on. An example might be the dormant seed stage in some plant species. These stage weight values should be set to 0.

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)

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)

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)

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")

cyca2_start <- start_input(cycamatrix2r, stage2 = c("SD", "P1", "P2"),
  value = c(500, 100, 200))

cyp2_dv <- density_input(cycamatrix2r, 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_eq <- stage_weight(cycamatrix2r,
  stage2 = c("SD", "P1", "SL", "D", "XSm", "Sm", "Md", "Lg", "XLg"),
  value = c(0, 1, 1, 1, 1, 1, 1, 1, 1))

cyca_proj <- projection3(mpm = cycamatrix2r, start_frame = cyca2_start,
  density = cyp2_dv, stage_weights = cyp_eq, times = 10)

summary(cyca_proj)
```

`start_input` *Create a Starting Vector for Population Projection*

Description

Function `start_input()` creates a data frame summarizing the non-zero elements of the start vector for use in population projection analysis via function `projection3()`.

Usage

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

Arguments

mpm	The lefkoMat object to be used in projection analysis.
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 lefkoSV, with four objects, which can be used as input in function projection3(). 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.

See Also

density_input()

projection3()

Examples

```
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
matvector <- c(0, 0, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)

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

lathvert <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "Volume88", repstracol = "FCODE88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframe, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, censor = TRUE)

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "Sd", "Sdl", "mat"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "rep", "rep", "Sdl"),
  stage1 = c("Sd", "rep", "Sd", "rep", "npr", "npr", "Sd"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, "mat"),
  eststage2 = c(NA, NA, NA, NA, NA, NA, "Sdl"),
  eststage1 = c(NA, NA, NA, NA, NA, NA, "NotAlive"),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, 0.345, 0.054, NA),
  type = c(1, 1, 1, 1, 3, 3, 1), type_t12 = c(1, 2, 1, 2, 1, 1, 1),
  stageframe = lathframe, historical = TRUE)

ehrlen3 <- rlefko3(data = lathvert, stageframe = lathframe, year = "all",
  stages = c("stage3", "stage2", "stage1"), supplement = lathsupp3,
  yearcol = "year2", indivcol = "individ")

ehrlen3mean <- lmean(ehrlen3)

e3m_sv <- start_input(ehrlen3mean, stage2 = "Sd", stage1 = "Sd", value = 1000)

lathproj <- projection3(ehrlen3, nreps = 5, times = 100, stochastic = TRUE,
  start_frame = e3m_sv)
```

subset_lM *Create New lefkoMat or lefkoMatList Object as Subset of Another*

Description

Function subset_lM() creates a new lefkoMat or lefkoMatList object from a subset of matrices in another lefkoMat or lefkoMatList object.

Usage

```
subset_lM(lM, mat_num = NA, pop = NA, patch = NA, year = NA)
```

Arguments

lM	The lefkoMat or lefkoMatList object to select matrices from.
mat_num	Either a single integer corresponding to the matrix to select within the labels element of lM, or a vector of such integers.
pop	The population designation for matrices to select. Only used if mat_num is not given.
patch	The patch designation for matrices to select. Only used if mat_num is not given.
year	The time *t* designation for matrices to select. Only used if mat_num is not given.

Value

A lefkoMat or lefkoMatList object composed of the matrices specified in the options. Note that, when applied to a lefkoMatList object, subsetting happens equivalently across all composite lefkoMat objects.

Notes

If mat_num is not provided, then at least one of pop, patch, or year must be provided. If at least two of pop, patch, and year are provided, then function subset_lM() will identify matrices as the intersection of provided inputs.

See Also

create_lM()

add_lM()

delete_lM()

Examples

```
# These matrices are of 9 populations of the plant species Anthyllis
# vulneraria, and were originally published in Davison et al. (2010) Journal
# of Ecology 98:255-267 (doi: 10.1111/j.1365-2745.2009.01611.x).

sizevector <- c(1, 1, 2, 3) # These sizes are not from the original paper
stagevector <- c("Sdl", "Veg", "SmFlo", "LFlo")
repvector <- c(0, 0, 1, 1)
obsvector <- c(1, 1, 1, 1)
matvector <- c(0, 1, 1, 1)
immvector <- c(1, 0, 0, 0)
propvector <- c(0, 0, 0, 0)
indataset <- c(1, 1, 1, 1)
binvec <- c(0.5, 0.5, 0.5, 0.5)

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

# POPN C 2003-2004
XC3 <- matrix(c(0, 0, 1.74, 1.74,
0.208333333, 0, 0, 0.057142857,
0.041666667, 0.076923077, 0, 0,
0.083333333, 0.076923077, 0.066666667, 0.028571429), 4, 4, byrow = TRUE)

# 2004-2005
XC4 <- matrix(c(0, 0, 0.3, 0.6,
0.32183908, 0.142857143, 0, 0,
0.16091954, 0.285714286, 0, 0,
0.252873563, 0.285714286, 0.5, 0.6), 4, 4, byrow = TRUE)

# 2005-2006
XC5 <- matrix(c(0, 0, 0.50625, 0.675,
0, 0, 0, 0.035714286,
0.1, 0.068965517, 0.0625, 0.107142857,
0.3, 0.137931034, 0, 0.071428571), 4, 4, byrow = TRUE)

# POPN E 2003-2004
XE3 <- matrix(c(0, 0, 2.44, 6.569230769,
0.196428571, 0, 0, 0,
0.125, 0.5, 0, 0,
0.160714286, 0.5, 0.133333333, 0.076923077), 4, 4, byrow = TRUE)

XE4 <- matrix(c(0, 0, 0.45, 0.646153846,
0.06557377, 0.090909091, 0.125, 0,
0.032786885, 0, 0.125, 0.076923077,
0.049180328, 0, 0.125, 0.230769231), 4, 4, byrow = TRUE)

XE5 <- matrix(c(0, 0, 2.85, 3.99,
0.083333333, 0, 0, 0,
0, 0, 0, 0,
```

```
0.416666667, 0.1, 0, 0.1), 4, 4, byrow = TRUE)

# POPN F 2003-2004
XF3 <- matrix(c(0, 0, 1.815, 7.058333333,
0.075949367, 0, 0.05, 0.083333333,
0.139240506, 0, 0, 0.25,
0.075949367, 0, 0, 0.083333333), 4, 4, byrow = TRUE)

XF4 <- matrix(c(0, 0, 1.233333333, 7.4,
0.223880597, 0, 0.111111111, 0.142857143,
0.134328358, 0.272727273, 0.166666667, 0.142857143,
0.119402985, 0.363636364, 0.055555556, 0.142857143), 4, 4, byrow = TRUE)

XF5 <- matrix(c(0, 0, 1.06, 3.372727273,
0.073170732, 0.025, 0.033333333, 0,
0.036585366, 0.15, 0.1, 0.136363636,
0.06097561, 0.225, 0.166666667, 0.272727273), 4, 4, byrow = TRUE)

# POPN G 2003-2004
XG3 <- matrix(c(0, 0, 0.245454545, 2.1,
0, 0, 0.045454545, 0,
0.125, 0, 0.090909091, 0,
0.125, 0, 0.090909091, 0.333333333), 4, 4, byrow = TRUE)

XG4 <- matrix(c(0, 0, 1.1, 1.54,
0.111111111, 0, 0, 0,
0, 0, 0, 0,
0.111111111, 0, 0, 0), 4, 4, byrow = TRUE)

XG5 <- matrix(c(0, 0, 0, 1.5,
0, 0, 0, 0,
0.090909091, 0, 0, 0,
0.545454545, 0.5, 0, 0.5), 4, 4, byrow = TRUE)

# POPN L 2003-2004
XL3 <- matrix(c(0, 0, 1.785365854, 1.856521739,
0.128571429, 0, 0, 0.010869565,
0.028571429, 0, 0, 0,
0.014285714, 0, 0, 0.02173913), 4, 4, byrow = TRUE)

XL4 <- matrix(c(0, 0, 14.25, 16.625,
0.131443299, 0.057142857, 0, 0.25,
0.144329897, 0, 0, 0,
0.092783505, 0.2, 0, 0.25), 4, 4, byrow = TRUE)

XL5 <- matrix(c(0, 0, 0.594642857, 1.765909091,
0, 0, 0.017857143, 0,
0.021052632, 0.018518519, 0.035714286, 0.045454545,
0.021052632, 0.018518519, 0.035714286, 0.068181818), 4, 4, byrow = TRUE)

# POPN O 2003-2004
XO3 <- matrix(c(0, 0, 11.5, 2.775862069,
0.6, 0.285714286, 0.333333333, 0.24137931,
```

```
0.04, 0.142857143, 0, 0,
0.16, 0.285714286, 0, 0.172413793), 4, 4, byrow = TRUE)

XO4 <- matrix(c(0, 0, 3.78, 1.225,
0.28358209, 0.171052632, 0, 0.166666667,
0.084577114, 0.026315789, 0, 0.055555556,
0.139303483, 0.447368421, 0, 0.305555556), 4, 4, byrow = TRUE)

XO5 <- matrix(c(0, 0, 1.542857143, 1.035616438,
0.126984127, 0.105263158, 0.047619048, 0.054794521,
0.095238095, 0.157894737, 0.19047619, 0.082191781,
0.111111111, 0.223684211, 0, 0.356164384), 4, 4, byrow = TRUE)

# POPN Q 2003-2004
XQ3 <- matrix(c(0, 0, 0.15, 0.175,
0, 0, 0, 0,
0, 0, 0, 0,
1, 0, 0, 0), 4, 4, byrow = TRUE)

XQ4 <- matrix(c(0, 0, 0, 0.25,
0, 0, 0, 0,
0, 0, 0, 0,
1, 0.666666667, 0, 1), 4, 4, byrow = TRUE)

XQ5 <- matrix(c(0, 0, 0, 1.428571429,
0, 0, 0, 0.142857143,
0.25, 0, 0, 0,
0.25, 0, 0, 0.571428571), 4, 4, byrow = TRUE)

# POPN R 2003-2004
XR3 <- matrix(c(0, 0, 0.7, 0.6125,
0.25, 0, 0, 0.125,
0, 0, 0, 0,
0.25, 0.166666667, 0, 0.25), 4, 4, byrow = TRUE)

XR4 <- matrix(c(0, 0, 0, 0.6,
0.285714286, 0, 0, 0,
0.285714286, 0.333333333, 0, 0,
0.285714286, 0.333333333, 0, 1), 4, 4, byrow = TRUE)

XR5 <- matrix(c(0, 0, 0.7, 0.6125,
0, 0, 0, 0,
0, 0, 0, 0,
0.333333333, 0, 0.333333333, 0.625), 4, 4, byrow = TRUE)

# POPN S 2003-2004
XS3 <- matrix(c(0, 0, 2.1, 0.816666667,
0.166666667, 0, 0, 0,
0, 0, 0, 0,
0, 0, 0, 0.166666667), 4, 4, byrow = TRUE)

XS4 <- matrix(c(0, 0, 0, 7,
0.333333333, 0.5, 0, 0,
```

```
0, 0, 0, 0,
0.333333333, 0, 0, 1), 4, 4, byrow = TRUE)

XS5 <- matrix(c(0, 0, 0, 1.4,
0, 0, 0, 0,
0, 0, 0, 0.2,
0.111111111, 0.75, 0, 0.2), 4, 4, byrow = TRUE)

mats_list <- list(XC3, XC4, XC5, XE3, XE4, XE5, XF3, XF4, XF5, XG3, XG4, XG5,
  XL3, XL4, XL5, XO3, XO4, XO5, XQ3, XQ4, XQ5, XR3, XR4, XR5, XS3, XS4, XS5)

yr_ord <- c(1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1,
  2, 3, 1, 2, 3)

pch_ord <- c(1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 4, 5, 5, 5, 6, 6, 6, 7, 7, 7,
  8, 8, 8, 9, 9, 9)

anth_lefkoMat <- create_lM(mats_list, anthframe, hstages = NA, historical = FALSE,
  poporder = 1, patchorder = pch_ord, yearorder = yr_ord)

smaller_anth_lM <- subset_lM(anth_lefkoMat, patch = c(1, 2, 3),
  year = c(1, 2))
```

`summary.lefkoCondMat` *Summary of Class "lefkoCondMat"*

Description

This function provides basic information summarizing the characteristics of conditional matrices derived from a `lefkoCondMat` object.

Usage

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

Arguments

`object`	An object of class `lefkoCondMat`.
`...`	Other parameters.

Value

A text summary of the object shown on the console, showing the number of historical matrices, as well as the number of conditional matrices nested within each historical matrix.

Examples

```
# Lathyrus example
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
matvector <- c(0, 0, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)

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

lathvert <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "Volume88", repstracol = "FCODE88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframe, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, censor = TRUE)

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "Sd", "Sdl", "mat"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "rep", "rep", "Sdl"),
  stage1 = c("Sd", "rep", "Sd", "rep", "npr", "npr", "Sd"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, "mat"),
  eststage2 = c(NA, NA, NA, NA, NA, NA, "Sdl"),
  eststage1 = c(NA, NA, NA, NA, NA, NA, "NotAlive"),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, 0.345, 0.054, NA),
  type = c(1, 1, 1, 1, 3, 3, 1), type_t12 = c(1, 2, 1, 2, 1, 1, 1),
  stageframe = lathframe, historical = TRUE)

ehrlen3 <- rlefko3(data = lathvert, stageframe = lathframe, year = "all",
  stages = c("stage3", "stage2", "stage1"), supplement = lathsupp3,
  yearcol = "year2", indivcol = "individ")

lathcondmats <- cond_hmpm(ehrlen3)
summary(lathcondmats)

# Cypripedium  example
data(cypdata)

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 = 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)

cypsupp3r <- supplemental(stage3 = c("SD", "SD", "P1", "P1", "P2", "P3", "SL",
    "D", "XSm", "Sm", "D", "XSm", "Sm", "mat", "mat", "mat", "SD", "P1"),
  stage2 = c("SD", "SD", "SD", "SD", "P1", "P2", "P3", "SL", "SL", "SL", "SL",
    "SL", "SL", "D", "XSm", "Sm", "rep", "rep"),
  stage1 = c("SD", "rep", "SD", "rep", "SD", "P1", "P2", "P3", "P3", "P3",
    "SL", "SL", "SL", "SL", "SL", "SL", "mat", "mat"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, NA, "D", "XSm", "Sm", "D", "XSm", "Sm",
    "mat", "mat", "mat", NA, NA),
  eststage2 = c(NA, NA, NA, NA, NA, NA, NA, "XSm", "XSm", "XSm", "XSm", "XSm",
    "XSm", "D", "XSm", "Sm", NA, NA),
  eststage1 = c(NA, NA, NA, NA, NA, NA, NA, "XSm", "XSm", "XSm", "XSm", "XSm",
    "XSm", "XSm", "XSm", "XSm", NA, NA),
  givenrate = c(0.1, 0.1, 0.2, 0.2, 0.2, 0.2, 0.25, NA, NA, NA, NA, NA, NA,
    NA, NA, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
    NA, 0.5, 0.5),
  type = c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3),
  type_t12 = c(1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1),
  stageframe = cypframe_raw, historical = TRUE)

cypmatrix3r <- rlefko3(data = cypraw_v1, stageframe = cypframe_raw,
  year = "all", patch = "all", stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added", "size1added"),
  supplement = cypsupp3r, yearcol = "year2", patchcol = "patchid",
  indivcol = "individ")

cypcondmats <- cond_hmpm(cypmatrix3r)

summary(cypcondmats)
```

---

summary.lefkoElas *Summarize lefkoElas Objects*

Description

Function `summary.lefkoElas()` summarizes `lefkoElas` objects. Particularly, it breaks down elasticity values by the kind of ahistorical and, if applicable, historical transition.

Usage

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

Arguments

`object`	A `lefkoElas` object.
`...`	Other parameters currently not utilized.

Value

A list composed of 2 data frames. The first, `hist`, is a data frame showing the summed elasticities for all 16 kinds of historical transition per matrix, with each column corresponding to each elasticity matrix in order. The second, `ahist`, is a data frame showing the summed elasticities for all 4 kinds of ahistorical transition per matrix, with each column corresponding to each elasticity matrix in order.

Examples

```
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
matvector <- c(0, 0, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)

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

lathvert <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "Volume88", repstracol = "FCODE88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframe, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, censor = TRUE)

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "Sd", "Sdl", "mat"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "rep", "rep", "Sdl"),
  stage1 = c("Sd", "rep", "Sd", "rep", "npr", "npr", "Sd"),
```

```
  eststage3 = c(NA, NA, NA, NA, NA, NA, "mat"),
  eststage2 = c(NA, NA, NA, NA, NA, NA, "Sdl"),
  eststage1 = c(NA, NA, NA, NA, NA, NA, "NotAlive"),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, 0.345, 0.054, NA),
  type = c(1, 1, 1, 1, 3, 3, 1), type_t12 = c(1, 2, 1, 2, 1, 1, 1),
  stageframe = lathframe, historical = TRUE)

lathsupp2 <- supplemental(stage3 = c("Sd", "Sdl", "Sd", "Sdl"),
  stage2 = c("Sd", "Sd", "rep", "rep"),
  givenrate = c(0.345, 0.054, NA, NA),
  multiplier = c(NA, NA, 0.345, 0.054),
  type = c(1, 1, 3, 3), stageframe = lathframe, historical = FALSE)

ehrlen3 <- rlefko3(data = lathvert, stageframe = lathframe, year = "all",
  stages = c("stage3", "stage2", "stage1"), supplement = lathsupp3,
  yearcol = "year2", indivcol = "individ")

ehrlen2 <- rlefko2(data = lathvert, stageframe = lathframe, year = "all",
  stages = c("stage3", "stage2"), supplement = lathsupp2,
  yearcol = "year2", indivcol = "individ")

ehrlen3elas <- elasticity3(ehrlen3)
ehrlen2elas <- elasticity3(ehrlen2)

summary(ehrlen3elas)
summary(ehrlen2elas)
```

`summary.lefkoLTRE` *Summarize lefkoLTRE Objects*

Description

Function `summary.lefkoLTRE()` summarizes `lefkoLTRE` objects. Particularly, it breaks down LTRE contributions by the kind of ahistorical and, if applicable, historical transition.

Usage

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

Arguments

`object`	A `lefkoLTRE` object.
`...`	Other parameters currently not utilized.

Value

A list of data frames. In all cases, the first data frame is one showing the positive, negative, and total contributions of elements in each LTRE contribution matrix. If not a SNA-LTRE, then there are an additional two (if deterministic) or four (if stochastic) data frames. If deterministic, then `hist_det` is a data frame showing the summed LTRE contributions for all 16 kinds of historical transition per matrix, with each column corresponding to each A matrix in order, followed by all summed positive and all summed negative contributions. Object `ahist_det` is a data frame showing the summed LTRE contributions for all four kinds of ahistorical transition per matrix, with order as before, followed by summed positive and summed negative contributions. If stochastic, then `hist_mean` and `hist_sd` are the summed LTRE contributions for the mean vital rates and variability in vital rates, respectively, according to all 16 historical transition types, followed by summed positive and negative contributions, and `ahist_mean` and `ahist_sd` are the equivalent ahistorical versions. The output for the SNA-LTRE also includes the logs of the deterministic lambda estimated through function `ltre3()`.

Examples

```
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
matvector <- c(0, 0, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)

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

lathvert <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "Volume88", repstracol = "FCODE88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframe, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, censor = TRUE)

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "Sd", "Sdl", "mat"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "rep", "rep", "Sdl"),
  stage1 = c("Sd", "rep", "Sd", "rep", "npr", "npr", "Sd"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, "mat"),
  eststage2 = c(NA, NA, NA, NA, NA, NA, "Sdl"),
  eststage1 = c(NA, NA, NA, NA, NA, NA, "NotAlive"),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, 0.345, 0.054, NA),
  type = c(1, 1, 1, 1, 3, 3, 1), type_t12 = c(1, 2, 1, 2, 1, 1, 1),
  stageframe = lathframe, historical = TRUE)
```

```
lathsupp2 <- supplemental(stage3 = c("Sd", "Sdl", "Sd", "Sdl"),
  stage2 = c("Sd", "Sd", "rep", "rep"),
  givenrate = c(0.345, 0.054, NA, NA),
  multiplier = c(NA, NA, 0.345, 0.054),
  type = c(1, 1, 3, 3), stageframe = lathframe, historical = FALSE)

ehrlen3 <- rlefko3(data = lathvert, stageframe = lathframe, year = "all",
  stages = c("stage3", "stage2", "stage1"), supplement = lathsupp3,
  yearcol = "year2", indivcol = "individ")

ehrlen2 <- rlefko2(data = lathvert, stageframe = lathframe, year = "all",
  stages = c("stage3", "stage2"), supplement = lathsupp2,
  yearcol = "year2", indivcol = "individ")

ehrlen3ltre <- ltre3(ehrlen3)
summary(ehrlen3ltre)
```

`summary.lefkoMat` *Summary of Class "lefkoMat"*

Description

A function to simplify the viewing of basic information describing the matrices produced through functions `flefko3()`, `flefko2()`, `rlefko3()`, `rlefko2()`, `aflefko2()`, `rleslie()`, and `fleslie()`.

Usage

```
## S3 method for class 'lefkoMat'
summary(object, colsums = TRUE, check_cycle = TRUE, ...)
```

Arguments

`object`	An object of class `lefkoMat`.
`colsums`	A logical value indicating whether column sums should be shown for U matrices, allowing users to check stage survival probabilities. Defaults to `TRUE`.
`check_cycle`	A logical value indicating whether to test matrices for stage discontinuities in the life cycle. Defaults to `TRUE`.
`...`	Other parameters.

Value

A summary of the object, showing the number of each type of matrix, the number of annual matrices, the number of estimated (non-zero) elements across all matrices and per matrix, the number of unique transitions in the dataset, the number of individuals, and summaries of the column sums of the survival-transition matrices. Stage discontinuities are also checked with function `cycle_check`. This function will also yield warnings if any survival-transition matrices include elements outside of the interval [0,1], if any fecundity matrices contain negative elements, and if any matrices include NA values.

Notes

Under the Gaussian and gamma size distributions, the number of estimated parameters may differ between the two ipm_method settings. Because the midpoint method has a tendency to incorporate upward bias in the estimation of size transition probabilities, it is more likely to yield non- zero values when the true probability is extremely close to 0. This will result in the summary.lefkoMat function yielding higher numbers of estimated parameters than the ipm_method = "CDF" yields in some cases.

Examples

```
data(cypdata)

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 = 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)

# Here we use supplemental() to provide overwrite and reproductive info
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", patch = "all", stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added"), supplement = cypsupp2r,
  yearcol = "year2", patchcol = "patchid", indivcol = "individ")
```

```
summary(cypmatrix2r)
```

`summary.lefkoMatList` *Summary of Class "lefkoMatList"*

Description

A function to simplify the viewing of basic information describing the bootstrapped matrices produced through functions `flefko3()`, `flefko2()`, `rlefko3()`, `rlefko2()`, `aflefko2()`, `rleslie()`, and `fleslie()` when using bootstrapped data input in format `hfvlist`.

Usage

```
## S3 method for class 'lefkoMatList'
summary(
  object,
  elem_summaries = FALSE,
  colsums = TRUE,
  check_cycle = TRUE,
  ...
)
```

Arguments

`object`	An object of class `lefkoMatList`.
`elem_summaries`	A logical value indicating whether to include summaries of all `lefkoMat` elements within the input `lefkoMatList` object. Defaults to `FALSE`.
`colsums`	If `elem_summaries = TRUE`, then this is a logical value indicating whether column sums should be shown for U matrices, allowing users to check stage survival probabilities. Defaults to `TRUE`.
`check_cycle`	If `elem_summaries = TRUE`, then this is a logical value indicating whether to test matrices for stage discontinuities in the life cycle. Defaults to `TRUE`.
`...`	Other parameters.

Value

A general summary of the `lefkoMatList` object, showing the number and type of `lefkoMat` objects included, the mean number of annual matrices per `lefkoMat` element, the mean size of each matrix, the mean number of estimated (non-zero) elements across all matrices and per matrix, the mean number of unique transitions in the bootstrapped datasets, the mean number of individuals, and summaries of the column sums of the survival-transition matrices across all `lefkoMat` objects.

if `elem_summaries = TRUE`, then `summary.lefkoMat{}` output is also shown for each `lefkoMat` object per entered `lefkoMatList` object.

Stage discontinuities may also be also checked across all `lefkoMat` objects with function `cycle_check`. This function will also yield warnings if any survival-transition matrices include elements outside of the interval [0,1], if any fecundity matrices contain negative elements, and if any matrices include NA values.

Notes

Under the Gaussian and gamma size distributions, the number of estimated parameters may differ between the two ipm_method settings. Because the midpoint method has a tendency to incorporate upward bias in the estimation of size transition probabilities, it is more likely to yield non- zero values when the true probability is extremely close to 0. This will result in the summary.lefkoMat function yielding higher numbers of estimated parameters than the ipm_method = "CDF" yields in some cases.

Examples

```
data(cypdata)

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 = 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)

cypraw_v1_boot <- bootstrap3(cypraw_v1, reps = 3)

# Here we use supplemental() to provide overwrite and reproductive info
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_boot <- rlefko2(data = cypraw_v1_boot, stageframe = cypframe_raw,
  year = "all", patch = "all", stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added"), supplement = cypsupp2r,
```

```
  yearcol = "year2", patchcol = "patchid", indivcol = "individ")

summary(cypmatrix2r_boot)
```

`summary.lefkoMod` *Summary of Class "lefkoMod"*

Description

A function to summarize objects of class `lefkoMod`. This function shows the best-fit models, summarizes the numbers of models in the model tables, shows the criterion used to determine the best-fit models, and provides some basic quality control information.

Usage

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

Arguments

`object`	An R object of class `lefkoMod` resulting from `modelsearch()`.
`...`	Other parameters currently not utilized.

Value

A summary of the object, showing the best-fit models for all vital rates, with constants of 0 or 1 used for unestimated models. This is followed by a summary of the number of models tested per vital rate, and a table showing the names of the parameters used to model vital rates and represent tested factors. At the end is a section describing the numbers of individuals and of individual transitions used to estimate each vital rate best-fit model, along with the accuracy of each binomial model.

Examples

```
# Lathyrus example
data(lathyrus)

sizevector <- c(0, 4.6, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 1, 2, 3, 4, 5, 6, 7, 8,
  9)
stagevector <- c("Sd", "Sdl", "Dorm", "Sz1nr", "Sz2nr", "Sz3nr", "Sz4nr",
  "Sz5nr", "Sz6nr", "Sz7nr", "Sz8nr", "Sz9nr", "Sz1r", "Sz2r", "Sz3r",
  "Sz4r", "Sz5r", "Sz6r", "Sz7r", "Sz8r", "Sz9r")
repvector <- c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1)
obsvector <- c(0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
matvector <- c(0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0)
indataset <- c(0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
```

```
binvec <- c(0, 4.6, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
  0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5)

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

lathvertln <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "lnVol88", repstracol = "Intactseed88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframeln,
  stagesize = "sizea", censorcol = "Missing1988", censorkeep = NA,
  NAas0 = TRUE, censor = TRUE)

lathvertln$feca2 <- round(lathvertln$feca2)
lathvertln$feca1 <- round(lathvertln$feca1)
lathvertln$feca3 <- round(lathvertln$feca3)

lathmodelsln2 <- modelsearch(lathvertln, historical = FALSE,
  approach = "mixed", suite = "main",
  vitalrates = c("surv", "obs", "size", "repst", "fec"), juvestimate = "Sdl",
  bestfit = "AICc&k", sizedist = "gaussian", fecdist = "poisson",
  indiv = "individ", patch = "patchid", year = "year2",
  year.as.random = TRUE, patch.as.random = TRUE, show.model.tables = TRUE,
  quiet = "partial")

summary(lathmodelsln2)
```

`summary.lefkoProj` *Summarize lefkoProj Objects*

Description

Function `summary.lefkoProj()` summarizes `lefkoProj` objects. Particularly, it breaks down the data frames provided in the `projection` element in ways meaningful for those running simulations.

Usage

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

Arguments

object	A lefkoProj 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

```
# Lathyrus example
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
```

```
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
matvector <- c(0, 0, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)

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

lathvert <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "Volume88", repstracol = "FCODE88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframe, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, censor = TRUE)

lathrepm <- matrix(0, 7, 7)
lathrepm[1, 6] <- 0.345
lathrepm[2, 6] <- 0.054

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "Sd", "Sdl"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "rep", "rep"),
  stage1 = c("Sd", "rep", "Sd", "rep", "all", "all"),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA),
  multiplier = c(NA, NA, NA, NA, 0.345, 0.054),
  type = c(1, 1, 1, 1, 3, 3), type_t12 = c(1, 2, 1, 2, 1, 1),
  stageframe = lathframe, historical = TRUE)

ehrlen3 <- rlefko3(data = lathvert, stageframe = lathframe,
  year = c(1989, 1990), stages = c("stage3", "stage2", "stage1"),
  repmatrix = lathrepm, supplement = lathsupp3, yearcol = "year2",
  indivcol = "individ")

lathproj <- projection3(ehrlen3, nreps = 5, stochastic = TRUE)
summary(lathproj)

# Cypripedium example
data(cypdata)

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 = 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)

cypsupp3r <- supplemental(stage3 = c("SD", "SD", "P1", "P1", "P2", "P3", "SL",
    "D", "XSm", "Sm", "D", "XSm", "Sm", "mat", "mat", "mat", "SD", "P1"),
  stage2 = c("SD", "SD", "SD", "SD", "P1", "P2", "P3", "SL", "SL", "SL", "SL",
    "SL", "SL", "D", "XSm", "Sm", "rep", "rep"),
  stage1 = c("SD", "rep", "SD", "rep", "SD", "P1", "P2", "P3", "P3", "P3",
    "SL", "SL", "SL", "SL", "SL", "SL", "mat", "mat"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, NA, "D", "XSm", "Sm", "D", "XSm", "Sm",
    "mat", "mat", "mat", NA, NA),
  eststage2 = c(NA, NA, NA, NA, NA, NA, NA, "XSm", "XSm", "XSm", "XSm", "XSm",
    "XSm", "D", "XSm", "Sm", NA, NA),
  eststage1 = c(NA, NA, NA, NA, NA, NA, NA, "XSm", "XSm", "XSm", "XSm", "XSm",
    "XSm", "XSm", "XSm", "XSm", NA, NA),
  givenrate = c(0.1, 0.1, 0.2, 0.2, 0.2, 0.2, 0.25, NA, NA, NA, NA, NA, NA,
    NA, NA, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
    NA, 0.5, 0.5),
  type = c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3),
  type_t12 = c(1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1),
  stageframe = cypframe_raw, historical = TRUE)

cypmatrix3r <- rlefko3(data = cypraw_v1, stageframe = cypframe_raw,
  year = "all", patch = "all", stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added", "size1added"),
  supplement = cypsupp3r, yearcol = "year2",
  patchcol = "patchid", indivcol = "individ")

cypstoch <- projection3(cypmatrix3r, nreps = 5, stochastic = TRUE)
summary(cypstoch, ext_time = TRUE)
```

`summary_hfv` *Summary of Classes "hfvdata" and "hfvlist*

Description

A function to simplify the viewing of basic information describing demographic data in historical vertical format (data frames of class `hfvdata`, or bootstrapped data frames in lists of class `hfvlist`).

Usage

```
summary_hfv(
  object,
  popid = "popid",
  patchid = "patchid",
  individ = "individ",
  year2id = "year2",
  full = FALSE,
  err_check = TRUE,
  ...
)
```

Arguments

object	An object of either class hfvdata or class hfvlist.
popid	A string denoting the name of the variable denoting population identity.
patchid	A string denoting the name of the variable denoting patch identity.
individ	A string denoting the name of the variable denoting individual identity.
year2id	A string denoting the name of the variable denoting the year in time *t*.
full	A logical value indicating whether to include basic data frame summary information in addition to hfvdata-specific summary information. Defaults to FALSE.
err_check	A logical value indicating whether to check for errors in stage assignment.
...	Other parameters.

Value

A summary of the object. If an object of class hfvdata is entered, then the first line of output shows the numbers of populations, patches, individuals, and time steps. If full = TRUE, then this is followed by a standard data frame summary of the hfv dataset. If err_check = TRUE, then a subset of the original data frame input as object is exported with only rows showing stage assignment issues.

If an object of class hfvlist in entered, then the first line of output shows the number of bootstrapped datasets. This is followed by lines showing the mean number of rows and variables per data frame, as well as the mean numbers of populations, patches, individuals, and years per data frame. This is followed by lines showing the total number of unique populations, patches, individuals, and years sampled across the bootstrapped data frames. If full = TRUE, then this is followed by standard data frame summaries of all bootstrapped data frames. If err_check = TRUE, then a data frame is output with all of the problem rows across all bootstrapped data frames, starting with a new variable giving the bootstrap number of origin.

Notes

Stage assignment issue identified by option err_check fall under two categories. First, all rows showing NoMatch as the identified stage for stage1, stage2, or stage3 are identified. Second, all rows showing stage1 = "NotAlive" and alive1 = 1, stage2 = "NotAlive" and alive2 = 1, or stage3 = "NotAlive" and alive3 = 1 are identified.

Examples

```
data(cypdata)

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 = 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)

summary_hfv(cypraw_v1)

cypraw_v1_boot <- bootstrap3(cypraw_v1, reps = 3)

summary_hfv(cypraw_v1_boot)
```

`supplemental` *Create a Data Frame of Supplemental Data for MPM Development*

Description

Function `supplemental()` provides all necessary supplemental data for matrix estimation. It allows the establishment of proxy rates, the entry of data to overwrite or offset existing rates, the identification of complete reproductive transitions, and the entry of rate multipliers. The function should be used to incorporate data that affects all matrices to be created. To edit MPMs after creation, use `edit_lM()` instead.

Usage

```
supplemental(
  historical = TRUE,
  stagebased = TRUE,
```

```
  agebased = FALSE,
  stageframe = NULL,
  stage3 = NULL,
  stage2 = NULL,
  stage1 = NULL,
  age2 = NULL,
  eststage3 = NULL,
  eststage2 = NULL,
  eststage1 = NULL,
  estage2 = NULL,
  givenrate = NULL,
  offset = NULL,
  multiplier = NULL,
  type = NULL,
  type_t12 = 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.
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*.

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.

Value

A data frame of class lefkoSD. This object can be used as input in flefko3(), flefko2(), rlefko3(), rlefko2(), and aflefko2().

Variables in this object include the following:

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.

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.

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), or a fecundity rate (2).

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.

Fecundity multiplier data supplied via the `supplemental()` function acts in the same way as non-zero entries supplied via a reproductive matrix, but gets priority in all matrix creations. Thus, in cases where fecundity multipliers are provided for the same function via the reproductive matrix and function `supplemental()`, the latter is used.

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

`edit_lM()`

Examples

```
# Lathyrus example
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
```

```
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
matvector <- c(0, 0, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)

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

lathvert <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
  juvcol = "Seedling1988", sizeacol = "Volume88", repstracol = "FCODE88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframe, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, censor = TRUE)

lathsupp3 <- supplemental(stage3 = c("Sd", "Sd", "Sdl", "Sdl", "Sd", "Sdl", "mat"),
  stage2 = c("Sd", "Sd", "Sd", "Sd", "rep", "rep", "Sdl"),
  stage1 = c("Sd", "rep", "Sd", "rep", "npr", "npr", "Sd"),
  eststage3 = c(NA, NA, NA, NA, NA, NA, "mat"),
  eststage2 = c(NA, NA, NA, NA, NA, NA, "Sdl"),
  eststage1 = c(NA, NA, NA, NA, NA, NA, "NotAlive"),
  givenrate = c(0.345, 0.345, 0.054, 0.054, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, 0.345, 0.054, NA),
  type = c(1, 1, 1, 1, 3, 3, 1), type_t12 = c(1, 2, 1, 2, 1, 1, 1),
  stageframe = lathframe, historical = TRUE)

ehrlen3 <- rlefko3(data = lathvert, stageframe = lathframe, year = "all",
  stages = c("stage3", "stage2", "stage1"), supplement = lathsupp3,
  yearcol = "year2", indivcol = "individ")

# Cypripedium example
data(cypdata)

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 = 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", "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", patch = "all", stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added"), supplement = cypsupp2r,
  yearcol = "year2", patchcol = "patchid", indivcol = "individ")
```

---

sup_skeleton *Create A Supplement Skeleton Data Frame*

Description

Create A Supplement Skeleton Data Frame

Usage

```
sup_skeleton(rows = NULL)
```

Arguments

rows	An integer giving the number of rows to include.

Value

A data frame with the format of a supplement, of class `lefkoSD`.

Examples

```
new_supp <- sup_skeleton(3)
new_supp
```

usher3 *Two-Parameter Usher Function*

Description

Function `usher3()` creates a vector of values produced by the two- parameter Usher function as applied with a user-specified time lag. The Usher function is given as $\phi_{t+1} = \phi_t/(1 + e^{\alpha n_t + \beta})$. Here, if no `separate_N` vector is provided, then $n_t = \phi_t$.

Usage

```
usher3(
  start_value,
  alpha,
  beta,
  time_steps = 100L,
  time_lag = 1L,
  pre0_subs = FALSE,
  pre0_value = 0,
  substoch = 0L,
  separate_N = NULL
)
```

Arguments

`start_value`	A positive number to start the return vector in time 0.
`alpha`	The alpha parameter in the two-parameter Usher function.
`beta`	The beta parameter in the two-parameter Usher function.
`time_steps`	The number of time steps to run the projection. Must be a positive integer.
`time_lag`	A positive integer denoting the number of time steps back for the value of phi in the two-parameter Usher function.
`pre0_subs`	A logical value indicating whether to use a number other than that given in `start_value` for values of phi lagged from times prior to time 0.
`pre0_value`	A positive number to use for phi lagged from times prior to time 0. Only used if `pre0_subs = TRUE`.
`substoch`	An integer value indicating the kind of substochasticity to use. Values include: `0`, no substochasticity enforced (the default); `1`, all numbers must be non-negative; and `2`, all numbers should be forced to the interval [0, 1].
`separate_N`	An optional numeric vector with values of N in each time, if phi is to be treated as different from N in the two-parameter model.

Value

A numeric vector of values showing values projected under the two- parameter Usher function.

Examples

```
trial_run1 <- usher3(1, alpha = -0.5, beta = 0.005)
plot(trial_run1)

trial_run2 <- usher3(1, alpha = 0.5, beta = 0.005)
plot(trial_run2)

trial_run3 <- usher3(1, alpha = -5, beta = 0.005)
plot(trial_run3)

trial_run4 <- usher3(1, alpha = 5, beta = 0.005)
plot(trial_run4)

trial_run5 <- usher3(1, alpha = -25, beta = 0.005)
plot(trial_run5)

trial_run6 <- usher3(1, alpha = 25, beta = 0.005)
plot(trial_run6)

used_Ns <- c(10, 15, 12, 14, 14, 150, 15, 1, 5, 7, 9, 14, 13, 16, 17, 19,
  25, 26)
trial_run7 <- usher3(1, alpha = -0.5, beta = 0.005, separate_N = used_Ns)
plot(trial_run7)
```

---

verticalize3 *Create Historical Vertical Data Frame from Horizontal Data Frame*

Description

Function `verticalize3()` returns a vertically formatted demographic data frame organized to create historical projection matrices, given a horizontally formatted input data frame. It also handles stage assignments if given an appropriate stageframe.

Usage

```
verticalize3(
  data,
  noyears,
  firstyear = 1,
  popidcol = 0,
  patchidcol = 0,
  individcol = 0,
  blocksize = NA,
  xcol = 0,
  ycol = 0,
  juvcol = 0,
  sizeacol,
```

```
  sizebcol = 0,
  sizeccol = 0,
  repstracol = 0,
  repstrbcol = 0,
  fecacol = 0,
  fecbcol = 0,
  indcovacol = 0,
  indcovbcol = 0,
  indcovccol = 0,
  aliveacol = 0,
  deadacol = 0,
  obsacol = 0,
  nonobsacol = 0,
  censorcol = 0,
  repstrrel = 1,
  fecrel = 1,
  stagecol = 0,
  stageassign = NA,
  stagesize = NA,
  censorkeep = 0,
  censorRepeat = FALSE,
  censor = FALSE,
  coordsRepeat = FALSE,
  spacing = NA,
  NAas0 = FALSE,
  NRasRep = FALSE,
  NOasObs = FALSE,
  prebreeding = TRUE,
  age_offset = 0,
  reduce = TRUE,
  a2check = FALSE,
  quiet = FALSE
)
```

Arguments

data	The horizontal data file. A valid data frame is required as input.
noyears	The number of years or observation occasions in the dataset. A valid integer is required as input.
firstyear	The first year or occasion of observation. Defaults to 1.
popidcol	A variable name or column number corresponding to the identity of the population for each individual.
patchidcol	A variable name or column number corresponding to the identity of the patch or subpopulation for each individual, if patches have been designated within populations.
individcol	A variable name or column number corresponding to the identity of each individual.

`blocksize` The number of variables corresponding to each occasion in the input dataset designated in `data`, if a set pattern of variables is used for each observation occasion in the data frame used as input. If such a pattern is not used, and all variable names are properly noted as character vectors in the other input variables, then this may be set to `NA`. Defaults to `NA`.

`xcol` A variable name(s) or column number(s) corresponding to the X coordinate of each individual, or of each individual at each occasion, in Cartesian space. Can refer to the only instance, the first instance, or all instances of X variables. In the last case, the values should be entered as a vector.

`ycol` A variable name(s) or column number(s) corresponding to the Y coordinate of each individual, or of each individual at each occasion, in Cartesian space. Can refer to the only instance, the first instance, or all instances of Y variables. In the last case, the values should be entered as a vector.

`juvcol` A variable name(s) or column number(s) that marks individuals in immature stages within the dataset. This function assumes that immature individuals are identified in this variable marked with a number equal to or greater than `1`, and that mature individuals are marked as `0` or `NA`. Can refer to the first instance, or all instances of these variables. In the latter case, the values should be entered as a vector.

`sizeacol` A variable name(s) or column number(s) corresponding to the size entry associated with the first year or observation occasion in the dataset. Can refer to the first instance, or all instances of these variables. In the latter case, the values should be entered as a vector. This variable should refer to the first size variable in the stageframe, unless `stagesize = "sizeadded"`.

`sizebcol` A second variable name(s) or column number(s) corresponding to the size entry associated with the first year or observation occasion in the dataset. Can refer to the first instance, or all instances of these variables. In the latter case, the values should be entered as a vector. This variable should refer to the second size variable in the stageframe, unless `stagesize = "sizeadded"`.

`sizeccol` A third variable name(s) or column number(s) corresponding to the size entry associated with the first year or observation occasion in the dataset. Can refer to the first instance, or all instances of these variables. In the latter case, the values should be entered as a vector. This variable should refer to the third size variable in the stageframe, unless `stagesize = "sizeadded"`.

`repstracol` A variable name(s) or column number(s) corresponding to the production of reproductive structures, such as flowers, associated with the first year or observation period in the input dataset. This can be binomial or count data, and is used to analyze the probability of reproduction. Can refer to the first instance, or all instances of these variables. In the latter case, the values should be entered as a vector.

`repstrbcol` A second variable name(s) or column number(s) corresponding to the production of reproductive structures, such as flowers, associated with the first year or observation period in the input dataset. This can be binomial or count data, and is used to analyze the probability of reproduction. Can refer to the first instance, or all instances of these variables. In the latter case, the values should be entered as a vector.

`fecacol` A variable name(s) or column number(s) denoting fecundity associated with the first year or observation occasion in the input dataset. This may represent egg counts, fruit counts, seed production, etc. Can refer to the first instance, or all instances of these variables. In the latter case, the values should be entered as a vector.

`fecbcol` A second variable name(s) or column number(s) denoting fecundity associated with the first year or observation occasion in the input dataset. This may represent egg counts, fruit counts, seed production, etc. Can refer to the first instance, or all instances of these variables. In the latter case, the values should be entered as a vector.

`indcovacol` A variable name(s) or column number(s) corresponding to an individual covariate to be used in analysis. Can refer to the only instance, the first instance, or all instances of these variables. In the last case, the values should be entered as a vector.

`indcovbcol` A variable name(s) or column number(s) corresponding to an individual covariate to be used in analysis. Can refer to the only instance, the first instance, or all instances of these variables. In the last case, the values should be entered as a vector.

`indcovccol` A second variable name(s) or column number(s) corresponding to an individual covariate to be used in analysis. Can refer to the only instance, the first instance, or all instances of these variables. In the last case, the values should be entered as a vector.

`aliveacol` Variable name(s) or column number(s) providing information on whether an individual is alive at a given occasion. If used, living status must be designated as binomial (living = `1`, dead = `0`). Can refer to the first instance of a living status variable in the dataset, or a full vector of all living status variables in temporal order.

`deadacol` Variable name(s) or column number(s) providing information on whether an individual is alive at a given occasion. If used, dead status must be designated as binomial (dead = `1`, living = `0`). Can refer to the first instance of a dead status variable in the dataset, or a full vector of all dead status variables in temporal order.

`obsacol` A variable name(s) or column number(s) providing information on whether an individual is in an observable stage at a given occasion. If used, observation status must be designated as binomial (observed = `1`, not observed = `0`). Can refer to the first instance of an observation status variable in the dataset, or a full vector of all observation status variables in temporal order.

`nonobsacol` A variable name(s) or column number(s) providing information on whether an individual is in an unobservable stage at a given occasion. If used, observation status must be designated as binomial (not observed = `1`, observed = `0`). Can refer to the first instance of a non-observation status variable in the dataset, or a full vector of all non-observation status variables in temporal order.

`censorcol` A variable name(s) or column number(s) corresponding to the first entry of a censor variable, used to distinguish between entries to use and entries not to use, or to designate entries with special issues that require further attention. Can refer to the first instance of a censor status variable in the dataset, or a full vector

of all censor status variables in temporal order. Can also refer to a single censor status variable used for the entire individual, if singlecensor = TRUE.

repstrrel — This is a scalar multiplier on variable repstrbcol to make it equivalent to repstracol. This can be useful if two reproductive status variables have related but unequal units, for example if repstracol refers to one-flowered stems while repstrbcol refers to two-flowered stems. Defaults to 1.

fecrel — This is a scalar multiplier on variable fecbcol to make it equivalent to fecacol. This can be useful if two fecundity variables have related but unequal units. Defaults to 1.

stagecol — Optional variable name(s) or column number(s) corresponding to life history stage at a given occasion. Can refer to the first instance of a stage identity variable in the dataset, or a full vector of all stage identity variables in temporal order.

stageassign — The stageframe object identifying the life history model being operationalized. Note that if stagecol is provided, then this stageframe is not used for stage designation.

stagesize — A variable name or column number describing which size variable to use in stage estimation. Defaults to NA, and can also take sizea, sizeb, sizec, sizeab, sizebc, sizeac, sizeabc, or sizeadded, depending on which size variable within the input dataset is chosen. Note that the variable(s) chosen should be presented in the order of the primary, secondary, and tertiary variables in the stageframe input with stageassign. For example, choosing sizeb assumes that this size is the primary variable in the stageframe.

censorkeep — The value of the censor variable identifying data to be included in analysis. Defaults to 0, but may take any value including NA. Note that if NA is the value to keep, then this function will alter all NAs to 0 values, and all other values to 1, treating 0 as the new value to keep.

censorRepeat — A logical value indicating whether the censor variable is a single column, or whether it repeats across occasion blocks. Defaults to FALSE.

censor — A logical variable determining whether the output data should be censored using the variable defined in censorcol. Defaults to FALSE.

coordsRepeat — A logical value indicating whether X and Y coordinates correspond to single X and Y columns. If TRUE, then each observation occasion has its own X and Y variables. Defaults to FALSE.

spacing — The spacing at which density should be estimated, if density estimation is desired and X and Y coordinates are supplied. Given in the same units as those used in the X and Y coordinates given in xcol and ycol. Defaults to NA.

NAas0 — If TRUE, then all NA entries for size and fecundity variables will be set to 0. This can help increase the sample size analyzed by modelsearch(), but should only be used when it is clear that this substitution is biologically realistic. Defaults to FALSE.

NRasRep — If TRUE, then will treat non-reproductive but mature individuals as reproductive during stage assignment. This can be useful when a MPM is desired without separation of reproductive and non-reproductive but mature stages of the same size. Only used if stageassign is set to a stageframe. Defaults to FALSE.

NOasObs	If TRUE, then will treat individuals that are interpreted as not observed in the dataset as though they were observed during stage assignment. This can be useful when a MPM is desired without separation of observable and unobservable stages. Only used if stageassign is set to a stageframe. Defaults to FALSE.
prebreeding	A logical term indicating whether the life history model is pre-breeding. If so, then 1 is added to all ages. Defaults to TRUE.
age_offset	A number to add automatically to all values of age at time *t*. Defaults to 0.
reduce	A logical variable determining whether unused variables and some invariant state variables should be removed from the output dataset. Defaults to TRUE.
a2check	A logical variable indicating whether to retain all data with living status at occasion *t*. Defaults to FALSE, in which case data for occasions in which the individual is not alive in time *t* is not retained. This option should be kept FALSE, except to inspect potential errors in the dataset.
quiet	A logical variable indicating whether to silence warnings. Defaults to FALSE.

Value

If all inputs are properly formatted, then this function will output a historical vertical data frame (class hfvdata), meaning that the output data frame will have three consecutive occasions of size and reproductive data per individual per row. This data frame is in standard format for all functions used in lefko3, and so can be used without further modification.

Variables in this data frame include the following:

rowid	Unique identifier for the row of the data frame.
popid	Unique identifier for the population, if given.
patchid	Unique identifier for patch within population, if given.
individ	Unique identifier for the individual.
year2	Year or time at occasion *t*.
firstseen	Occasion of first observation.
lastseen	Occasion of last observation.
obsage	Observed age in occasion *t*, assuming first observation corresponds to age = 0.
obslifespan	Observed lifespan, given as lastseen - firstseen + 1.
xpos1, xpos2, xpos3	X position in Cartesian space in occasions *t*-1, *t*, and *t*+1, respectively, if provided.
ypos1, ypos2, ypos3	Y position in Cartesian space in occasions *t*-1, *t*, and *t*+1, respectively, if provided.
sizea1, sizea2, sizea3	Main size measurement in occasions *t*-1, *t*, and *t*+1, respectively.
sizeb1, sizeb2, sizeb3	Secondary size measurement in occasions *t*-1, *t*, and *t*+1, respectively.
sizec1, sizec2, sizec3	Tertiary measurement in occasions *t*-1, *t*, and *t*+1, respectively.

`size1added`, `size2added`, `size3added`
: Sum of primary, secondary, and tertiary size measurements in occasions *t*-1, *t*, and *t*+1, respectively.

`repstra1`, `repstra2`, `repstra3`
: Main numbers of reproductive structures in occasions *t*-1, *t*, and *t*+1, respectively.

`repstrb1`, `repstrb2`, `repstrb3`
: Secondary numbers of reproductive structures in occasions *t*-1, *t*, and *t*+1, respectively.

`repstr1added`, `repstr2added`, `repstr3added`
: Sum of primary and secondary reproductive structures in occasions *t*-1, *t*, and *t*+1, respectively.

`feca1`, `feca2`, `feca3`
: Main numbers of offspring in occasions *t*-1, *t*, and *t*+1, respectively.

`fecb1`, `fecb2`, `fecb3`
: Secondary numbers of offspring in occasions *t*-1, *t*, and *t*+1, respectively.

`fec1added`, `fec2added`, `fec3added`
: Sum of primary and secondary fecundity in occasions *t*-1, *t*, and *t*+1, respectively.

`censor1`, `censor2`, `censor3`
: Censor state values in occasions *t*-1, *t*, and *t*+1, respectively.

`juvgiven1`, `juvgiven2`, `juvgiven3`
: Binomial variable indicating whether individual is juvenile in occasions *t*-1, *t*, and *t*+1. Only given if `juvcol` is provided.

`obsstatus1`, `obsstatus2`, `obsstatus3`
: Binomial observation state in occasions *t*-1, *t*, and *t*+1, respectively.

`repstatus1`, `repstatus2`, `repstatus3`
: Binomial reproductive state in occasions *t*-1, *t*, and *t*+1, respectively.

`fecstatus1`, `fecstatus2`, `fecstatus3`
: Binomial offspring production state in occasions *t*-1, *t*, and *t*+1, respectively.

`matstatus1`, `matstatus2`, `matstatus3`
: Binomial maturity state in occasions *t*-1, *t*, and *t*+1, respectively.

`alive1`, `alive2`, `alive3`
: Binomial state as alive in occasions *t*-1, *t*, and *t*+1, respectively.

`density`
: Radial density of individuals per unit designated in `spacing`. Only given if `spacing` is not NA.

Notes

In some datasets on species with unobservable stages, observation status (`obsstatus`) might not be inferred properly if a single size variable is used that does not yield sizes greater than 0 in all cases in which individuals were observed. Such situations may arise, for example, in plants when leaf number is the dominant size variable used, but individuals occasionally occur with inflorescences but no leaves. In this instances, it helps to mark related variables as `sizeb` and `sizec`, because observation status will be interpreted in relation to all 3 size variables. Further analysis can then

utilize only a single size variable, of the user's choosing. Similar issues can arise in reproductive status (repstatus).

Juvenile designation should only be used when juveniles fall outside of the size classification scheme used in determining stages. If juveniles are to be size classified along the size spectrum that adults also fall on, then it is best to treat juveniles as mature but not reproductive.

Warnings that some individuals occur in state combinations that do not match any stages in the stageframe used to assign stages are common when first working with a dataset. Typically, these situations can be identified as NoMatch entries in stage3, although such entries may crop up in stage1 and stage2, as well. In rare cases, these warnings will arise with no concurrent NoMatch entries, which indicates that the input dataset contained conflicting state data at once suggesting that the individual is in some stage but is also dead. The latter is removed if the conflict occurs in occasion *t* or *t*-1, as only living entries are allowed in time *t* and time *t*-1 may involve living entries as well as non-living entries immediately prior to birth.

Care should be taken to avoid variables with negative values indicating size, fecundity, or reproductive or observation status. Negative values can be interpreted in different ways, typically reflecting estimation through other algorithms rather than actual measured data. Variables holding negative values can conflict with data management algorithms in ways that are difficult to predict.

Unusual errors (e.g. "Error in .pfj...") may occur in cases where the variables are improperly passed, where seemingly numeric variables include text, or where the blocksize is improperly set.

Density estimation is performed as a count of individuals alive and within the radius specified in spacing of the respective individual at some point in time.

If a censor variable is included for each monitoring occasion, and the blocksize option is set, then the user must set censorRepeat = TRUE in order to censor the correct transitions. Failing this step will likely lead to the loss of a large portion of the data as all data for entire individuals will be excluded.

Examples

```
# Lathyrus example using blocksize - when repeated patterns exist in variable
# order
data(lathyrus)

sizevector <- c(0, 100, 13, 127, 3730, 3800, 0)
stagevector <- c("Sd", "Sdl", "VSm", "Sm", "VLa", "Flo", "Dorm")
repvector <- c(0, 0, 0, 0, 0, 1, 0)
obsvector <- c(0, 1, 1, 1, 1, 1, 0)
matvector <- c(0, 0, 1, 1, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 100, 11, 103, 3500, 3800, 0.5)

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

lathvert <- verticalize3(lathyrus, noyears = 4, firstyear = 1988,
  patchidcol = "SUBPLOT", individcol = "GENET", blocksize = 9,
```

```
  juvcol = "Seedling1988", sizeacol = "Volume88", repstracol = "FCODE88",
  fecacol = "Intactseed88", deadacol = "Dead1988",
  nonobsacol = "Dormant1988", stageassign = lathframe, stagesize = "sizea",
  censorcol = "Missing1988", censorkeep = NA, censor = TRUE)

# Cypripedium example using partial repeat patterns with blocksize and part
# explicit variable name cast
data(cypdata)

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 = cypdata, noyears = 6, firstyear = 2004,
  patchidcol = "patch", individcol = "plantid", blocksize = 4,
  sizeacol = "Inf2.04", sizebcol = "Inf.04", sizeccol = "Veg.04",
  repstracol = c("Inf.04", "Inf.05", "Inf.06", "Inf.07", "Inf.08", "Inf.09"),
  repstrbcol = c("Inf2.04", "Inf2.05", "Inf2.06", "Inf2.07", "Inf2.08", "Inf2.09"),
  fecacol = "Pod.04", stageassign = cypframe_raw, stagesize = "sizeadded",
  NAas0 = TRUE, NRasRep = TRUE)
```

---

vrm_import *Import Vital Rate Model Factor Values for Function-based MPM Development*

Description

Function vrm_import() builds a skeleton list holding data frames and vectors that can be used to import coefficient values for the factors of the vital rate models used to build function-based MPMs or run function-based projections.

Usage

```
vrm_import(
  years = NULL,
  patches = c(1),
  groups = c(0),
```

```
  interactions = FALSE,
  zi = FALSE,
  cat.indcova = NULL,
  cat.indcovb = NULL,
  cat.indcovc = NULL,
  dist.sizea = "gaussian",
  dist.sizeb = "constant",
  dist.sizec = "constant",
  dist.fec = "gaussian",
  trunc.sizea = FALSE,
  trunc.sizeb = FALSE,
  trunc.sizec = FALSE,
  trunc.fec = FALSE,
  use.juv = FALSE
)
```

Arguments

years	A numeric vector of the years or times at time t to be modeled.
patches	A string or numeric vector of the patch names to be modeled.
groups	An integer vector of stage groups to be modeled. Defaults to a vector with a single element with value 0.
interactions	A logical value indicating whether to include two-way interactions between main effects (TRUE), or only main effects (FALSE). Defaults to FALSE.
zi	A logical value indicating whether to include coefficients for the binomial components of zero-inflation models. Defaults to FALSE.
cat.indcova	If individual covariate a is categorical, then this term should equal a string vector of the names of the categories. Defaults to NULL, in which case individual covariate a is either not used or is numeric.
cat.indcovb	If individual covariate b is categorical, then this term should equal a string vector of the names of the categories. Defaults to NULL, in which case individual covariate b is either not used or is numeric.
cat.indcovc	If individual covariate c is categorical, then this term should equal a string vector of the names of the categories. Defaults to NULL, in which case individual covariate c is either not used or is numeric.
dist.sizea	A string value giving the distribution of the variable coding primary size. Can equal "none", "gamma", "gaussian", "poisson", "negbin", or "constant". Defaults to "gaussian".
dist.sizeb	A string value giving the distribution of the variable coding secondary size. Can equal "none", "gamma", "gaussian", "poisson", "negbin", or "constant". Defaults to "constant".
dist.sizec	A string value giving the distribution of the variable coding tertiary size. Can equal "none", "gamma", "gaussian", "poisson", "negbin", or "constant". Defaults to "constant".

dist.fec — A string value giving the distribution of the variable coding fecundity. Can equal "none", "gamma", "gaussian", "poisson", or "negbin". Defaults to "gaussian".

trunc.sizea — A logical value indicating whether the distribution of the primary size variable should be zero-truncated. Defaults to FALSE. Currently only works with the Poisson and negative binomial distributions.

trunc.sizeb — A logical value indicating whether the distribution of the secondary size variable should be zero-truncated. Defaults to FALSE. Currently only works with the Poisson and negative binomial distributions.

trunc.sizec — A logical value indicating whether the distribution of the tertiary size variable should be zero-truncated. Defaults to FALSE. Currently only works with the Poisson and negative binomial distributions.

trunc.fec — A logical value indicating whether the distribution of the fecundity variable should be zero-truncated. Defaults to FALSE. Currently only works with the Poisson and negative binomial distributions.

use.juv — A logical value indicating whether to utilize juvenile vital rates. If FALSE, then all juvenile vital rates will be set to constant distributions. Defaults to FALSE.

Value

A list of class vrm_input, with up to 13 elements including:

vrm_frame — A data frame holding the main slope coefficients for the linear vital rate models.

year_frame — A data frame holding the main slope coefficients for the year at time t terms in the linear vital rate models.

patch_frame — A data frame holding the main slope coefficients for the patch terms in the linear vital rate models.

group2_frame — A data frame holding the main slope coefficients for the stage group terms in time *t* in the linear vital rate models.

group1_frame — A data frame holding the main slope coefficients for the stage group terms in time *t*-1 in the linear vital rate models.

dist_frame — A data frame giving the distributions of all variables, including primary, secondary, and tertiary size, and fecundity. Some variables begin as constant.

indcova2_frame — A data frame holding the main slope coefficients for the categorical individual covariate a terms in time *t* in the linear vital rate models.

indcova1_frame — A data frame holding the main slope coefficients for the categorical individual covariate a terms in time *t*-1 in the linear vital rate models.

indcovb2_frame — A data frame holding the main slope coefficients for the categorical individual covariate b terms in time *t* in the linear vital rate models.

indcovb1_frame — A data frame holding the main slope coefficients for the categorical individual covariate b terms in time *t*-1 in the linear vital rate models.

indcovc2_frame — A data frame holding the main slope coefficients for the categorical individual covariate c terms in time *t* in the linear vital rate models.

indcovc1_frame — A data frame holding the main slope coefficients for the categorical individual covariate c terms in time *t*-1 in the linear vital rate models.

st_frame	A data frame holding values of sigma or theta for use in Gaussian or negative binomial response terms, respectively.

The first element, called vrm_frame, is a data frame with the following 18 variables:

main_effect_1	The main effect for which coefficients are to be entered.
main_1_defined	A more natural explanation of main_effect_1.
main_effect_2	If given, then indicates another effect in a two-way interaction with main_effect_1.
main_2_defined	A more natural explanation of main_effect_2.
surv	A vector of coefficients for the factors in the model of adult survival.
obs	A vector of coefficients for the factors in the model of adult observation status.
sizea	A vector of coefficients for the factors in the model of adult primary size.
sizeb	A vector of coefficients for the factors in the model of adult secondary size.
sizec	A vector of coefficients for the factors in the model of adult tertiary size.
repst	A vector of coefficients for the factors in the model of adult reproductive status.
fec	A vector of coefficients for the factors in the model of adult fecundity.
jsurv	A vector of coefficients for the factors in the model of juvenile survival.
jobs	A vector of coefficients for the factors in the model of juvenile observation status.
jsize	A vector of coefficients for the factors in the model of juvenile primary size.
jsizeb	A vector of coefficients for the factors in the model of juvenile secondary size.
jsizec	A vector of coefficients for the factors in the model of juvenile tertiary size.
jrepst	A vector of coefficients for the factors in the model of juvenile reproductive status, for individuals maturing in the current time step.
jmat	A vector of coefficients for the factors in the model of maturity status, for individuals capable of maturing at the current time step.
sizea_zi	A vector of coefficients for the factors in the binomial component of the zero-inflated model of adult primary size, if zero-inflated models are being used.
sizeb_zi	A vector of coefficients for the factors in the binomial component of the zero-inflated model of adult secondary size, if zero-inflated models are being used.
sizec_zi	A vector of coefficients for the factors in the binomial component of the zero-inflated model of adult tertiary size, if zero-inflated models are being used.
fec_zi	A vector of coefficients for the factors in the binomial component of the zero-inflated model of fecundity, if zero-inflated models are being used.
jsizea_zi	A vector of coefficients for the factors in the binomial component of the zero-inflated model of juvenile primary size, if zero-inflated models are being used.
jsizeb_zi	A vector of coefficients for the factors in the binomial component of the zero-inflated model of juvenile secondary size, if zero-inflated models are being used.
jsizec_zi	A vector of coefficients for the factors in the binomial component of the zero-inflated model of juvenile tertiary size, if zero-inflated models are being used.

Notes

All coefficients across all data frames are initially set to `0`. After using this function to create the skeleton list, all relevant coefficient values should be set to non-zero values equal to the respective slope from the appropriate linear model, and any vital rate model to be used should have its distribution set to `"binom"`, `"gaussian"`, `"gamma"`, `"poisson"`, or `"negbin"`. Unused vital rates should be set to `"constant"`, and the first element of the correspoding column in `vrm_frame` (corresponding to the y-intercept) should be set to the constant value to utilize (generally `1`). If no values are manually edited, then function-based MPM generator functions will not be able to generate valid MPMs.

Users should never change the labels or the order of terms in the data frames and vectors produced via this function, nor should they ever changes the names of core list elements in the `vrm_input` object. Doing so will result either in fatal errors or erroneous matrix calculations.

Using the `vrm_import()` approach to building function-based MPMs requires attention to the stageframe. Although no `hfv_data` object needs to be input, stages for which vital rates are to be estimated via models parameterized with coefficients provided via function `vrm_import()` should be marked as occurring within the dataset, while stages for which the provided coefficients should not be used should be marked as not occurring within the dataset.

Coefficients added to zero-inflation models can only be added to primary size, secondary size, tertiary size, fecundity, and the juvenile versions of primary, secondary, and tertiary size. Care must be taken to include zero- inflated coefficients only for variables without size-truncated distributions. Adding such terms will result in fatal errors during matrix creation.

Examples

```
data(lathyrus)

sizevector <- c(0, 100, 0, 1, 7100)
stagevector <- c("Sd", "Sdl", "Dorm", "ipm", "ipm")
repvector <- c(0, 0, 0, 1, 1)
obsvector <- c(0, 1, 0, 1, 1)
matvector <- c(0, 0, 1, 1, 1)
immvector <- c(1, 1, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0)
indataset <- c(0, 1, 1, 1, 1)
binvec <- c(0, 100, 0.5, 1, 1)
comments <- c("Dormant seed", "Seedling", "Dormant", "ipm adult stage",
  "ipm adult stage")
lathframeipm <- sf_create(sizes = sizevector, stagenames = stagevector,
  repstatus = repvector, obsstatus = obsvector, propstatus = propvector,
  immstatus = immvector, matstatus = matvector, comments = comments,
  indataset = indataset, binhalfwidth = binvec, ipmbins = 100, roundsize = 3)

lathsupp2 <- supplemental(stage3 = c("Sd", "Sdl", "Sd", "Sdl"),
  stage2 = c("Sd", "Sd", "rep", "rep"),
  givenrate = c(0.345, 0.054, NA, NA),
  multiplier = c(NA, NA, 0.345, 0.054),
  type = c(1, 1, 3, 3), stageframe = lathframeipm, historical = FALSE)

lath_vrm <- vrm_import(years = c(1988:1990), zi = TRUE, dist.fec = "negbin",
```

```
  use.juv = TRUE)

lath_vrm$vrm_frame$surv[1] <- 2.32571
lath_vrm$vrm_frame$surv[2] <- 0.00109
lath_vrm$vrm_frame$obs[1] <- 2.230
lath_vrm$vrm_frame$sizea[1] <- 164.0695
lath_vrm$vrm_frame$sizea[2] <- 0.6211
lath_vrm$vrm_frame$fec[1] <- 1.517
lath_vrm$vrm_frame$fec_zi[1] <- 6.252765
lath_vrm$vrm_frame$fec_zi[2] <- -0.007313
lath_vrm$vrm_frame$jsurv[1] <- 1.03
lath_vrm$vrm_frame$jobs[1] <- 10.390
lath_vrm$vrm_frame$jsizea[1] <- 3.0559
lath_vrm$vrm_frame$jsizea[2] <- 0.8482

lath_vrm$year_frame$fec[c(1:3)] <- c(-0.41749627, 0.51421684, -0.07964038)
lath_vrm$year_frame$fec_zi[c(1:3)] <- c(3.741475e-07, -7.804715e-08,
  -2.533755e-07)
lath_vrm$year_frame$sizea[c(1:3)] <- c(96.3244, -240.8036, 144.4792)
lath_vrm$year_frame$jobs[c(1:3)] <- c(-0.7459843, 0.6118826, -0.9468618)
lath_vrm$year_frame$jsizea[c(1:3)] <- c(0.5937962, 1.4551236, -2.0489198)

lath_vrm$dist_frame$dist[2] <- "binom"
lath_vrm$dist_frame$dist[9] <- "binom"

lath_vrm$st_frame[3] <- 503.6167
lath_vrm$st_frame[7] <- 0.2342114
lath_vrm$st_frame[10] <- 5.831

lath_vrm$vrm_frame$sizeb[1] <- 1
lath_vrm$vrm_frame$sizec[1] <- 1
lath_vrm$vrm_frame$repst[1] <- 1
lath_vrm$vrm_frame$jsizeb[1] <- 1
lath_vrm$vrm_frame$jsizec[1] <- 1
lath_vrm$vrm_frame$jrepst[1] <- 1
lath_vrm$vrm_frame$jmatst[1] <- 1

lathmat2_importipm <- flefko2(stageframe = lathframeipm,
  modelsuite = lath_vrm, supplement = lathsupp2, reduce = FALSE)

summary(lathmat2_importipm)
```

Index