Package: matreex (via r-universe)

May 29, 2026

Title matreex: Simulation IPM for forest dynamic depending on climatic variables from Kunstler et al (2020)

Version 0.4.0

Description The goal of this package is to run integrated projection models of tree species in single or multi-specific density dependence context. The simulations return the size distribution dynamics along time. These models can be completed with different harvest and disturbance models and be runned untile equilibrium.

License MIT + file LICENSE

Depends R (>= 3.5.0)

Imports checkmate, cli, dplyr, here, Matrix, purrr, rlang, statmod, stats, tidyr

Suggests lifecycle, covr, ggplot2, knitr, latex2exp, rmarkdown, testthat (>= 3.0.0), magick, pdftools

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

LazyDataCompression bzip2

URL https://github.com/gowachin/matreex

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Config/pak/sysreqs libicu-dev

Repository https://inrae.r-universe.dev

Date/Publication 2024-03-04 13:47:37 UTC

RemoteUrl https://forge.inrae.fr/lessem/rpackages/matreex

RemoteRef HEAD

RemoteSha 3b685b6fd13436231799271af6608953d7e62c95

Contents

`climate_species` *Species climate optimum dataset.*

Description

Climate defined per species from Kunstler et al. 2021. Each species had its IPM integrated for 3 climates :

1 hot edge

2 optimum climate for the species

3 cold edge

Usage

```
climate_species
```

Format

An object of class `data.frame` with 81 rows and 11 columns.

Details

Climate was defined from sgdd and wai variables.

Source

https://doi.org/10.1111/1365-2745.13533

`climatic` *Get climatic values in matreex class object*

Description

Get climatic values in matreex class object

Usage

```
climatic(x)
```

Arguments

`x`	matreex class object. Used on ipm and species at this time.

`correction` *Add correction states to a system*

Description

Apply correction to a matrix by setting to 0 all right and bottom values

Usage

```
correction(x, correction = "none")
```

Arguments

`x`	an object that require a correction addition
`correction`	Type of correction, match between cut and none (default). chr.

`def_disturb` *Default disturbance function*

Description

Default disturbance function

Usage

```
def_disturb(x, species, disturb = NULL, ...)
```

Arguments

`x`	population state distribution at time t
`species`	The species class object of interest to get mesh and RDIcoef values from. RDIcoef is a one line dataframe with RDI coefficient for one species.
`disturb`	Disturbance parameters. Highly depend on the disturbance impact parameters given to the species.
`...`	Default disturbance function does not require

`def_harv` *Default population harvest*

Description

Constant rate harvest of 0.06 percent per year (check if harvest frequence is 1 in forest object).

Usage

```
def_harv(x, species, ...)
```

Arguments

`x`	population state at time t
`species`	ignored
`...`	Variables used in this case of Uneven harvest **ct** is the vector to compute BA with x (ct = Buildct(mesh, SurfEch))

Details

This function is the default function for `harvest_fun` function of a species and always takes arguments `x`, `species` plus specific argument from different harvest models..

Value

Distribution of population to harvest. Values are between 0 (null harvest) and Xi.

See Also

Other functions that defines harvest rules for a species.: `Even_harv()`, `Uneven_harv()`, `species()`

`def_init` *Default population initialization*

Description

The population will initiate with a random distribution to match a basal area of 1.

Usage

```
def_init(mesh, SurfEch = 0.03)
```

Arguments

`mesh`	all possible states of a population, based on an IPM. Minimal and maximal values are respectively U and L, for a total number of m states.
`SurfEch`	Value of plot size surface in m^2

Details

This function is the default function for `init_pop` function of a species and always takes arguments `mesh` and `SurfEch`.

See Also

Other functions for initiating species population during simulation: `def_initBA()`, `def_init_even()`, `def_init_k()`, `species()`

`def_init_even` *Default even population initialization*

Description

The population will initiate with only individual in the 5 first cells of the mesh.

Usage

```
def_init_even(mesh, SurfEch = 0.03)
```

Arguments

`mesh`	all possible states of a population, based on an IPM. Minimal and maximal values are respectively U and L, for a total number of m states.
`SurfEch`	Value of plot size surface in m^2

See Also

Other functions for initiating species population during simulation: `def_initBA()`, `def_init_k()`, `def_init()`, `species()`

`def_init_k` *Init population with precise distribution*

Description

Init population with precise distribution

Usage

```
def_init_k(x)
```

Arguments

`x`	Distribution to draw systematically. This distribution should be composed of values in `[0, Inf]` values with a sum superior to 0. This is the distribution per hectare and not for the sampled plot.

Details

The resulting function will check if the provided vector is the same length as mesh.

Value

Function similar def_init but with no random effect anymore.

See Also

Other functions for initiating species population during simulation: `def_initBA()`, `def_init_even()`, `def_init()`, `species()`

`def_initBA` *Init population at BA*

Description

This function modify the def_init function to start at a given BA with the same process of random distribution.

Usage

```
def_initBA(BA = 1, fun = c("def_init", "def_init_even"))
```

Arguments

`BA`	Basal area targeted. This single value must be above 0 but can be very close (minimal accepted value is 1e-10)
`fun`	Function to modify, single chr in choices.

Value

Function similar to def_init

See Also

Other functions for initiating species population during simulation: `def_init_even()`, `def_init_k()`, `def_init()`, `species()`

`delay` *Add delay states to a system*

Description

Add delay states to an existing pop_state class object or an ipm class object.

Usage

```
delay(x, delay = 0)
```

Arguments

`x`	an object that require a delay addition
`delay`	the number of time delay to add. single int. Its possible to give negative values to remove delay from previously delayed objects. However final delay must be positive (0 is acceptable).

Details

This function is a method that call `delay.ipm` internal function.

`delay.dgCMatrix` *Delay dgCMatrix*

Description

Adding a topleft corner to a matrix filled with 0.

Usage

```
## S3 method for class 'dgCMatrix'
delay(x, delay = 0)
```

Arguments

`x`	an object that require a delay addition
`delay`	the number of time delay to add. single int. Its possible to give negative values to remove delay from previously delayed objects. However final delay must be positive (0 is acceptable).

Examples

```
x <- new("dtCMatrix", i = c(0L, 1L, 2L, 1L, 2L, 2L), p = c(0L, 3L,  5L, 6L),
         Dim = c(3L, 3L), x = c(0.1, 0.2, 0.3, 0.1, 0.2, 0.1), uplo = "L", diag = "N")
x <- delay(x, 2)
delay(x, -1)
```

`delay.dsCMatrix` *Delay dsCMatrix*

Description

Adding a topleft corner to a matrix filled with 0.

Usage

```
## S3 method for class 'dsCMatrix'
delay(x, delay = 0)
```

Arguments

`x`	an object that require a delay addition
`delay`	the number of time delay to add. single int. Its possible to give negative values to remove delay from previously delayed objects. However final delay must be positive (0 is acceptable).

Examples

```
x <- new("dtCMatrix", i = c(0L, 1L, 2L, 1L, 2L, 2L), p = c(0L, 3L,  5L, 6L),
         Dim = c(3L, 3L), x = c(0.1, 0.2, 0.3, 0.1, 0.2, 0.1), uplo = "L", diag = "N")
x <- delay(x, 2)
delay(x, -1)
```

`delay.dtCMatrix` *Delay dtCMatrix*

Description

Adding a topleft corner to a matrix filled with 0.

Usage

```
## S3 method for class 'dtCMatrix'
delay(x, delay = 0)
```

Arguments

`x`	an object that require a delay addition
`delay`	the number of time delay to add. single int. Its possible to give negative values to remove delay from previously delayed objects. However final delay must be positive (0 is acceptable).

Examples

```
x <- new("dtCMatrix", i = c(0L, 1L, 2L, 1L, 2L, 2L), p = c(0L, 3L,  5L, 6L),
         Dim = c(3L, 3L), x = c(0.1, 0.2, 0.3, 0.1, 0.2, 0.1), uplo = "L", diag = "N")
x <- delay(x, 2)
delay(x, -1)
```

`disturb_coef` *Species disturbance parameters.*

Description

Species disturbance parameters.

Usage

```
disturb_coef
```

Format

An object of class `data.frame` with 64 rows and 10 columns.

Source

https://doi.org/10.1111/gcb.16630

`disturb_fun` *Disturbance function*

Description

Disturbance function

Usage

```
disturb_fun(x, species, disturb = NULL, ...)
```

Arguments

`x`	population state distribution at time t
`species`	The species class object of interest to get mesh and RDIcoef values from. RDIcoef is a one line dataframe with RDI coefficient for one species.
`disturb`	Disturbance parameters. Highly depend on the disturbance impact parameters given to the species.
`...`	Not used in this case. **qmd** Forest Quadratic Mean Diameter

Details

Delayed mesh cells takes the value of the minimal mesh size.

`disturb_fun_mixt` *Disturbance function*

Description

Disturbance function

Usage

```
disturb_fun_mixt(x, species, disturb = NULL, ...)
```

Arguments

`x`	population state distribution at time t
`species`	The species class object of interest to get mesh and RDIcoef values from. RDIcoef is a one line dataframe with RDI coefficient for one species.
`disturb`	Disturbance parameters. Highly depend on the disturbance impact parameters given to the species.
`...`	Not used in this case. **qmd** Forest Quadratic Mean Diameter

Details

Delayed mesh cells takes the value of the minimal mesh size.

`Even_harv` *Perform the harvest Even*

Description

Perform the harvest Even

Usage

```
Even_harv(x, species, ...)
```

Arguments

`x`	population state distribution at time t
`species`	The species class object of interest to get mesh and RDIcoef values from. RDIcoef is a one line dataframe with RDI coefficient for one species.
`...`	Variables used in this case of Uneven harvest **targetKg** is the target Kg **targetRDI** is the target RDI **ct** is the vector to compute BA with x (ct = Buildct(mesh, SurfEch)) **SurfEch** Value of plot size surface in ha

Details

Even harvest is done at the stand level with the getPcutEven function to acomodate with the multi-specific case.

See Also

Other functions that defines harvest rules for a species.: `Uneven_harv()`, `def_harv()`, `species()`

`fit_data` *Species fited models dataset.*

Description

Reformated fitted models from Kunstler et al. 2021. Each species is fitted for growth, survival and recruitment along climatic conditions based on sgdd and wai. All objects were modified with the oldfit2fit function from this package.

Usage

```
fit_species

lag_species

fit_Abies_alba

fit_Acer_campestre

fit_Acer_pseudoplatanus

fit_Alnus_glutinosa

fit_Betula

fit_Carpinus_betulus

fit_Fagus_sylvatica

fit_Fraxinus_excelsior

fit_Juniperus_thurifera

fit_Larix_decidua

fit_Picea_abies

fit_Pinus_halepensis

fit_Pinus_nigra

fit_Pinus_pinaster

fit_Pinus_pinea

fit_Pinus_sylvestris
```

`fit_Pinus_uncinata`

`fit_Populus_tremula`

`fit_Prunus_padus`

`fit_Quercus_faginea`

`fit_Quercus_ilex`

`fit_Quercus_petraea`

`fit_Quercus_pubescens`

`fit_Quercus_pyrenaica`

`fit_Quercus_robur`

`fit_Quercus_suber`

`fit_Salix_caprea`

Format

An object of class `character` of length 27.

An object of class `numeric` of length 27.

An object of class `fit_sgr` of length 4.

An object of class `fit_sgr` of length 4.

An object of class `fit_sgr` of length 4.

An object of class `fit_sgr` of length 4.

An object of class `fit_sgr` of length 4.

An object of class `fit_sgr` of length 4.

An object of class `fit_sgr` of length 4.

An object of class `fit_sgr` of length 4.

An object of class `fit_sgr` of length 4.

An object of class `fit_sgr` of length 4.

An object of class `fit_sgr` of length 4.

An object of class `fit_sgr` of length 4.

An object of class `fit_sgr` of length 4.

An object of class `fit_sgr` of length 4.

An object of class `fit_sgr` of length 4.

An object of class `fit_sgr` of length 4.

An object of class `fit_sgr` of length 4.

An object of class `fit_sgr` of length 4.

An object of class `fit_sgr` of length 4.

An object of class `fit_sgr` of length 4.

An object of class `fit_sgr` of length 4.

An object of class `fit_sgr` of length 4.

An object of class `fit_sgr` of length 4.

An object of class `fit_sgr` of length 4.

An object of class `fit_sgr` of length 4.

An object of class `fit_sgr` of length 4.

An object of class `fit_sgr` of length 4.

Details

The fit_species dataset contains the vector of species stored with the package. The lag_species dataset contains the vector of default delay for each species.

The lag was computed by Kunstler and missing values for Betula, Juniperus_thurifera, Prunus_padus, Quercus_faginea and Quercus_pyrenaica are set with the mean of other species. More details is given in https://github.com/gowachin/matreex/issues/10

Source

https://doi.org/10.1111/1365-2745.13533

`fit_sgr` *Create a new fitted models from*

Description

Species are defined by an IPM which is a transition matrix from size between t and t+1, recruitment and harvest functions. Each species has these items defined for a given climate. An additionnal vector of harvest parameers is required with minimal size to harvest (dth), size above wich harvest is constant (dha).

Usage

```
fit_sgr(
  sv_params,
  sv_family,
  gr_params,
  gr_sigma,
  rec_params,
  species,
  max_dbh,
  delay
)
```

Arguments

`sv_params`	Named vector of survival parameters fitted for this species and climatic condition. Minimal parameters are intercept and size.
`sv_family`	Family list contain the details of the models used when fitting by functions such as glm. Only the inverse of the link function (linkinv) is used in the model for now. list.
`gr_params`	Named vector of growth parameters fitted for this species and climatic condition. Minimal parameters are intercept and size.
`gr_sigma`	Standard deviation of the residuals for the growth fitted model.
`rec_params`	Named vector of growth parameters fitted for this species and climatic condition. Minimal parameters are intercept, BATOTSP and BATOTNonSP.
`species`	Name of the species to run simulation on. Single char.
`max_dbh`	Maximum diameter of the fitted dataset. Single dbl.
`delay`	Delay for the species. Single dbl.

`forest` *Create a new forest for simulation*

Description

A forest is a group of one of multiple species to silumate along time using the IPM defined for each species and harvest rules.

Usage

```
forest(
  species = list(),
  harv_rules = c(Pmax = 0.25, dBAmin = 3, freq = 1, alpha = 1),
  favoured_sp = c()
)
```

Arguments

`species`	List of species created with matreex package.
`harv_rules`	Vector for harvest rules at the scale of the forest. **Pmax** maximum proportion of BAcut / BA **dBAmin** the minimum BA to perform cut **freq** Frequence at which the harvest will be executed. **alpha** parameter for uneven harvest, allowing to target more cuts for abundant species or not. See Harvesting vignette.
`favoured_sp`	Logical named vector to tell if species are favoured during Uneven harvesting or not. If not NULL, the species names should be the same as in the species list.

`get_maxdbh` *Get the max_dbh of a fitted species*

Description

Get the max_dbh of a fitted species

Usage

```
get_maxdbh(x)
```

Arguments

`x`	matreex class object. Used on fit_sgr for now.

`get_step_IPM` *Extract an IPM matrix*

Description

Extract the IPM matrix needed during simulation for a given BA and optionally climate.

Usage

```
get_step_IPM(x, ...)
```

Arguments

`x`	IPM or mu_gr class object.
`...`	Variables used depending on the class of x. **BA** Total basal area to get the IPM for during simulation. **climate** Climate for which the IPM is needed. **sim_corr** Simulation correction applied to the IPM. "cut" or "none" **IsSurv** Does this step IPM require survival. If missing or NULL, the value will be taken from x$info object

Details

Methods are set for IPM objects and mu_gr. Only the second one require to give the climate and simulation correction to apply to the matrix.

`make_IPM` *Build IPM for a given species and climate*

Description

Integrate IPM for growth and survival function at a specific climate for a species on a basal area variation.

Usage

```
make_IPM(
  species,
  climate,
  clim_lab,
  fit,
  mesh = c(m = 700, L = 90, U = 1500),
  BA = 0:100,
  correction = c("none", "constant", "ceiling", "sizeExtremes"),
  level = 420,
  diag_tresh = 50,
  midbin_tresh = 25,
  mid_level = 5,
  year_delta = 1,
  IsSurv = TRUE,
  delay = NULL,
  verbose = FALSE
)
```

Arguments

`species`	The species names to be registered in the object
`climate`	Named vector of the environmental variables used in the fitted model. Was data_plot_pred before.
`clim_lab`	Label for climatic used. This values will be matched when simulating multiple species together.
`fit`	Fitted model for growth and survival of the species and climate given. Functions will depend on size and basal area.
`mesh`	vector of mesh variables. m is the number of bins, L is the minimum size and U the maximum size. h will be defined in the function as $h <- (U - L)/m$.
`BA`	Vector of basal area to integrate on. Integrating on 0 is important so use it. Integrating above 200 is absurd.
`correction`	Correction to apply to the IPM matrix for eviction. Choices none (default), constant, ceiling and sizeExtremes.
`level`	Number of point to use for integration in a cell during Gauss-Legendre integration. This value will be divided by 3 since size t is integrated at level = 3 and size t+1 at level = level/3. single int (default 420).

`diag_tresh`	Threshold for Gauss-Legendre integration, which a distance to the diagonal. Number of cell integrated is the number of cell for which size t+1 - size t is inferior to this threshold. single dbl (default 50).
`midbin_tresh`	Number of cells external to the GL integration to integrate with the mid bin method.
`mid_level`	Number of point to use for integration in a cell during mid bin integration.
`year_delta`	Number of year between 2 obersavtion when using this model. default 1, single int. NOTE : value for dev usage only !
`IsSurv`	Adding survival to the IPM. Set to FALSE is useful to test for eviction of the model. TRUE by default.
`delay`	the number of time delay to add. By default, it will use the delay stored in the `fit` object provided, but can be provided.
`verbose`	Print message. FALSE by default

Details

The check between climate variables and fitted variable will assert if all variables in the model are provided expect variables derived from "size" (size, size2, logsize), "intercept" and "BATOTcomp". An error will be triggered if the climate variable is missing.

One can desactivate each kind of integration with some treshold values. A negative value in diag_tresh (ex: -1) will cancel the Gauss-Legendre integration and a midbin_tresh null value (ex: 0) will cancel the midbin integration.

`make_mu_gr` *Build IPM for a given species and climate*

Description

Integrate IPM for growth and survival function at a specific climate for a species on a basal area variation.

Usage

```
make_mu_gr(
  species,
  fit,
  climate = NULL,
  mesh = c(m = 700, L = 90, U = 1500),
  BA = 0:200,
  correction = c("none", "constant", "ceiling", "sizeExtremes"),
  stepMu = 0.001,
  level = c(3, 140),
  diag_tresh = 50,
  midbin_tresh = 25,
  mid_level = 5,
```

```
  year_delta = 1,
  IsSurv = TRUE,
  verbose = FALSE
)
```

Arguments

`species`	The species names to be registered in the object
`fit`	Fitted model for growth and survival of the species and climate given. Functions will depend on size and basal area.
`climate`	Climate table for the species. Optionnal if the species is defined in the package. The format is climatic variable in column and different climate by row. An example is in the package with `cliamte_species`.
`mesh`	vector of mesh variables. m is the number of bins, L is the minimum size and U the maximum size. h will be defined in the function as $h < -(U - L)/m$.
`BA`	Vector of basal area to integrate on. Integrating on 0 is important so use it. Integrating above 200 is absurd.
`correction`	Correction to apply to the IPM matrix for eviction. Choices constant (default), ceiling, sizeExtremes and none.
`stepMu`	Step between each mu in the species range. This value has effect simulation. Default is 1e-3. Values below 1e-5 takes longer than classical matrix integration.
`level`	Number of point to use for integration in a cell during Gauss-Legendre integration. This value will be divided by 3 since size t is integrated at level = 3 and size t+1 at level = level/3. single int (default 420).
`diag_tresh`	Threshold for Gauss-Legendre integration, which a distance to the diagonal. Number of cell integrated is the number of cell for which size t+1 - size t is inferior to this threshold. single dbl (default 50).
`midbin_tresh`	Number of cells external to the GL integration to integrate with the mid bin method.
`mid_level`	Number of point to use for integration in a cell during mid bin integration.
`year_delta`	Number of year between 2 obersavtion when using this model. default 1, single int. NOTE : value for dev usage only !
`IsSurv`	Adding survival to the IPM. Set to FALSE is useful to test for eviction of the model. TRUE by default.
`verbose`	Print message. FALSE by defaul

Details

The check between climate variables and fitted variable will assert if all variables in the model are provided expect variables derived from "size" (size, size2, logsize), "intercept" and "BATOTcomp". An error will be triggered if the climate variable is missing.

One can desactivate each kind of integration with some treshold values. A negative value in diag_tresh (ex: -1) will cancel the Gauss-Legendre integration and a midbin_tresh null value (ex: 0) will cancel the midbin integration.

This is a working function to test faster integration but it integrate a ba value

---

`matreex` *matreex*

Description

This package is developpped for LESSEM internal users only. It allow to run simulation from IPM and NFI datasets.

Details

Fill the package description !

`old_fit2fit` *load old fitted models*

Description

load old fitted models

Usage

```
old_fit2fit(species, path = here(), replicat = 42, mean = FALSE)
```

Arguments

`species`	Name of the species to run simulation on. Single char.
`path`	Place to save the resulting file. Single Char.
`replicat`	Numeric for the simulation to select. By default, the 42th.
`mean`	Should the return element be a mean of all models or a single value. FALSE by default, TRUE will ignore replicat argument

---

`old_ipm2forest` *Dev function to read old ipm*

Description

Read old ipm and put them in the new class format.

Usage

```
old_ipm2forest(sp_name, climatic = 1, path = here(), replicat = 42)
```

Arguments

`sp_name`	Name of the species to add in a single forest object. char.
`climatic`	Vector of climatic situations to run on. IPM must exist for each one or else this climatic value will be skipped. int.
`path`	Place to save the resulting file. Single Char.
`replicat`	Numeric for the simulation to select. By default, the 42th.

`old_ipm2species` *Dev function to read old ipm*

Description

Read old ipm and put them in the new species class format.

Usage

```
old_ipm2species(
  species,
  climatic = 1,
  path = here(),
  replicat = 42,
  harvest = def_harv,
  disturb = def_disturb,
  init_pop = def_init,
  delay = 0
)
```

Arguments

`species`	Name of the species to load. Single char.
`climatic`	Numeric that coded old climatic state. int.
`path`	Place to load the previous IPM file. Single Char.
`replicat`	Numeric for the model to select. By default, the 42th.
`harvest`	Function to impact the population with harvest rule. Argument must be pop. Should return a population state as it's take it in input, with less population than before. Unless you want zombie trees.
`disturb`	Function to impact the population with possibles disturbances. Extra care is needed to give this function all needed parameters/ Default is `def_disturb`.
`init_pop`	Function to initiate the population at simulation start. Arguments must be `mesh` and `SurfEch`. Using the arguments is not mandatory, it's most usefull when creating random population.
`delay`	Number of year delay between the recruitment of an individual and it's inclusion in the IPM. This will enlarge the IPM and add sub diagonal values of 1. See `delay`.

`QMD` *get Quadratic Mean Diameter*

Description

Compute the quadratic mean diameter for a given size distribution.

Usage

```
QMD(size, n)
```

Arguments

`size`	Size class vector of the following distribution. dbl vector
`n`	population state distribution at time t. dbl vector.

Details

Both input share the same length.

`rdi_coef` *Species Rdi intercept and slope.*

Description

Species Rdi intercept and slope.

Usage

```
rdi_coef
```

Format

An object of class `data.frame` with 28 rows and 4 columns.

Details

Missing values for Quercus faginea from french NFI

Source

Personnal contact with Patrick Vallet

`sim_deter_forest` *Deterministic Tree population simulation*

Description

Simulate a population size state during $[1, t_{lim}]$ times. The state at time $t+1$ is dependent on state at time t and the projection matrix (IPM). The simulation will run until t_{lim} and if the equilibrium is not reached, it will continue. Only simulation in $[1, t_{lim}]$, and equilibrium state are returned.

Usage

```
sim_deter_forest(
  Forest,
  tlim = 3000,
  equil_dist = 250,
  equil_diff = 1,
  equil_time = 10000,
  harvest = c("default", "Uneven", "Even", "Favoured_Uneven"),
  targetBA = 20,
  targetRDI = 0.9,
  targetKg = 0.9,
  final_harv = 100,
  climate = NULL,
  disturbance = NULL,
  correction = "none",
  SurfEch = 0.03,
  verbose = FALSE
)
```

Arguments

`Forest`	Group of species that each contains IPM for deterministic transition for Z_t state in a population to Z_{t+1} state. A species is also defined with recruitment and harvest functions, please see `species` for more information.
`tlim`	Number of simulation iterations (years) in the future. single int.
`equil_dist`	Number of last n time for which the range difference should not exceed `equil_diff` during the equilibrium research. single int.
`equil_diff`	Difference threshold of the basal area under which equilibrium is assumed. single real.
`equil_time`	Total maximum time simulation allowed in equilibrium research. Must be higher or equal to tlim and equil_dist. single int.
`harvest`	Choice of harvest rules between default, Uneven and Even. This indicate what settings will be used. See Details.
`targetBA`	BA value per ha that is targeted when using uneven harvesting. Single numeric in m^2.

`targetRDI`	RDI value that is targeted when using even harvesting. RDI is the ratio between the number of trees and the maximum number of trees given the self-thinning boundary for the corresponding mean diameter and species.
`targetKg`	Kg value that is targeted when using even harvesting. Kg is the ratio between mean quadratic diameter of killed trees and mean quadratic diameter of trees before harvesting.
`final_harv`	Final harvest time used when `harvest` is set to "Even". This parameter drives the final cut time for even stands.
`climate`	Optional, climate matrix if climate variation along time is needed. Climate variation rely on species created with mu_gr class objects. This matrix require as many rows as time steps until equil_time. If the climate does not variate, a single row can given and will be reused.
`disturbance`	**[Experimental]** parameter.
`correction`	Choice of correction of the IPM between `"none"` (default) and `"cut"`. The second option set the last column to 0 in the IPM so that no individual can grow outside of the defines classes.
`SurfEch`	Value of plot size surface in ha
`verbose`	Print message. FALSE by default

Details

At each iteration, the basal area is evaluated to select the corresponding IPM matrix.

Basic simulations input are illustrated in the main vignette. The harvesting scenario and theory is explained in the harvesting vignette.

Value

Data.frame with long tidyverse format : a row for each observation and a column per variable. Columns are listed below, some may contains NA values, as for example species when there is a non-specific variable (BA).

species Name of the species.

var Variable of interest

time Time step of the simulation. If the equilibrium is the last time in `tlim` input, this time will occur twice in the table.

mesh Mesh class number, from 1 to n class.

size Size corresponding to the mesh class.

equil Logical if this time step is the equilibrium or last step of simulation

value Numeric values of the variables.

The variables are :

n Distribution of density by mesh along time per ha.

N Sum of density per ha. (colSums for n)

BAsp Basal area of the population per ha and species

BAstand Basal area of the population per ha and species when excluding size class below dth. See Harvesting vignette.

h Distribution of harvest density by mesh along time per ha.

H Sum of harvested density per ha. (colSums for h)

`sim_rdikg` *Compute RDI and Kg from simulations output*

Description

Compute RDI and Kg from simulations output

Usage

```
sim_rdikg(sim, rdi_c = NULL)
```

Arguments

`sim`	simulation output table.
`rdi_c`	rdic coefficients for all species. Named vector. If NULL (default, the values are taken from matreex::rdi_coef)

`sp_name` *List the species in matreex class object*

Description

List the species in matreex class object

Usage

```
sp_name(x)
```

Arguments

`x`	matreex class object. Used on ipm and species at this time.

`sp_rec` *sp recruit*

Description

Get species recruitment function

Usage

```
sp_rec(x, climatic, regional = FALSE)
```

Arguments

`x`	Species to get the recruitment function from
`climatic`	Climate vector is needed for mu_gr object to build the corresponding recruitment function.
`regional`	TRUE/FALSE if we want to use a regional BA for fecundity.

`species` *Create a new species for simulation*

Description

Species are defined by an IPM which is a transition matrix from size between t and t+1, recruitment and harvest functions (see Details). Each species has these items defined for a given climate. An additionnal vector of harvest parameters is required with minimal size to harvest (dth), size above wich harvest is constant (dha).

Usage

```
species(
  IPM,
  init_pop = def_init,
  harvest_fun = def_harv,
  disturb_fun = def_disturb,
  harv_lim = c(dth = 175, dha = 575, hmax = 1),
  rdi_coef = NULL,
  disturb_coef = NULL,
  type = c("Undefined", "Broadleaf", "Coniferous")
)
```

Arguments

`IPM`	ipm class object from the matreex package.
`init_pop`	Function to initiate the population at simulation start. Arguments must be `mesh` and `SurfEch`. Using the arguments is not mandatory, it's most useful when creating random population.
`harvest_fun`	Function to impact the population with harvest rule. Argument must be `x`, `species`, `...`,
`disturb_fun`	Function to impact the population with possibles disturbances. Extra care is needed to give this function all needed parameters. Default is `def_disturb`.
`harv_lim`	Limits of harvest for a population size distribution. **dth** minimum diameter at which we cut the size distribution **dha** harvest diameter **hmax** maximum harvest rate for a size class
`rdi_coef`	Coefficient for RDI curve used in even harvest. The model require the intercept and slope.
`disturb_coef`	Species coefficient for disturbance reaction. These values and names are highly dependent on the disturbance function. `BAtarget`, `ct` and `t`. Should return a population state as it's take it in input, with less population than before. Unless you want zombie trees. It represent the distribution of the population to harvest
`type`	Type of the tree, choosing between "Broadleaf" and "Coniferous". This value is only used during biotic disturbance with a specific disturb_fun. This is experimental

Details

A species is defined by an IPM that is an integrated prediction matrix for growth and survival functions of the species. Since the species has other functions defined, they are accessible and editable.

`init_pop` Function to initiate a new population. Default is `def_init`.

`recruit_fun` Function that give a distribution for recruits. The default is defined from models associated with the IPM (`x$IPM$rec`) but it's possible to replace it. For example you can nullify the recruitment to simulate extinction.

`harvest_fun` Function that give harvest density distribution when an harvest event occurs (this frequence is set at the forest scale.). The default function is `def_harv` with a constant harvest rate of 0.6 percent. Other functions are `Uneven_harv` and `Even_harv`.

See Also

Other functions for initiating species population during simulation: `def_initBA()`, `def_init_even()`, `def_init_k()`, `def_init()`

Other functions that defines harvest rules for a species.: `Even_harv()`, `Uneven_harv()`, `def_harv()`

`summary.ipm` *IPM summary*

Description

Additionnal argument :

fline skip the first line. FALSE by default.

Usage

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

Arguments

`object`	Ipm object.
`...`	Dots are dots.

`summary.mu_gr` *mu_gr summary*

Description

This function is not written yet.

Usage

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

Arguments

`object`	Ipm object.
`...`	Dots are dots.

`summary.species` *species summary*

Description

Additionnal argument :

sum_ipm Print IPM summary. TRUE by default.

Usage

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

Arguments

`object`	Ipm object.
`...`	Dots are dots.

`tree_type` *Species type.*

Description

Species type.

Usage

```
tree_type
```

Format

An object of class `data.frame` with 28 rows and 2 columns.

Source

Maxime Jaunatre

`Uneven_harv` *Perform the harvest Uneven*

Description

Perform the harvest Uneven

Usage

```
Uneven_harv(x, species, ...)
```

Arguments

`x`	population state distribution at time t
`species`	The species class object of interest to get mesh and harv_lim values from.
`...`	Variables used in this case of Uneven harvest **targetBAcut** Basal Area to cut. **ct** is the vector to compute BA with x (ct = Buildct(mesh, SurfEch))

See Also

Other functions that defines harvest rules for a species.: `Even_harv()`, `def_harv()`, `species()`

Index