Package 'mizer'

January 27, 2019

Title Multi-Species sIZE Spectrum Modelling in R

Type Package

Description A set of classes and methods to set up and run multi-species, trait based and community size spectrum ecological models, focused on the marine environment.

Maintainer Gustav Delius <`gustav.delius@york.ac.uk`>

Version 1.0.1

License GPL-3

Imports deSolve, ggplot2, grid, methods, plyr, progress, Rcpp, reshape2

LinkingTo Rcpp

Depends R (>= 3.1)

Suggests testthat, roxygen2, knitr, shiny, shinyBS

Collate 'MizerParams-class.R' 'MizerSim-class.R' 'selectivity_funcs.R' 'project.R' 'help.R' 'project_methods.R' 'summary_methods.R' 'plots.R' 'wrapper_functions.R' 'data.R' 'RcppExports.R'

VignetteBuilder knitr

RoxygenNote 6.1.1

NeedsCompilation yes

Author Finlay Scott [aut, cph],
Julia Blanchard [aut, cph],
Ken Andersen [aut, cph],
Gustav Delius [ctb, cre],
Richard Southwell [ctb]

Repository CRAN

Date/Publication 2019-01-27 22:42:08 UTC

R topics documented:

addSpecies 3

display_frames . . . 4
getBiomass . . . 5
getBiomassFrame . . . 6
getCommunitySlope . . . 7
getEGrowth . . . 8
getEReproAndGrowth . . . 9
getESpawning . . . 10
getFeedingLevel . . . 11
getFMort . . . 13
getFMortGear . . . 14
getM2 . . . 16
getM2Background . . . 18
getMeanMaxWeight . . . 19
getMeanWeight . . . 20
getN . . . 21
getPhiPrey . . . 22
getPredRate . . . 23
getProportionOfLargeFish . . . 24
getRDD . . . 25
getRDI . . . 26
getSSB . . . 27
getSSBFrame . . . 28
getYield . . . 29
getYieldGear . . . 30
getZ . . . 30
get_initial_n . . . 32
inter . . . 32
knife_edge . . . 33
log_breaks . . . 33
mizer . . . 34
MizerParams . . . 34
MizerParams-class . . . 36
MizerSim . . . 38
MizerSim-class . . . 39
NS_species_params . . . 40
NS_species_params_gears . . . 40
plot,MizerSim,missing-method . . . 41
plotBiomass . . . 42
plotFeedingLevel . . . 43
plotFMort . . . 44
plotGrowthCurves . . . 45
plotM2 . . . 46
plotSpectra . . . 47
plotYield . . . 49
plotYieldGear . . . 50
project . . . 51
project_methods . . . 53
retune_abundance . . . 54

setBackground 55
set_community_model 56
set_scaling_model 58
set_trait_model 60
sigmoid_length 63
steady 64
summary,MizerParams-method 64
summary,MizerSim-method 65
wrapper_functions 65

Index **67**

`addSpecies` *Add more species into an ecosystem with background species.*

Description

Takes a MizerParams object and adds an additional species with given parameters to the ecosystem.

Usage

```
addSpecies(params, ...)

## S4 method for signature 'MizerParams'
addSpecies(params, species_params, SSB = NA,
  rfac = 10, effort = 0)
```

Arguments

`params`	A mizer params object for the original system.
`...`	Other arguments (unused)
`species_params`	The species parameters of the new species we want to add to the system.
`SSB`	The spawning stock biomass of the new species. If not provided, the abundance of the new species will be chosen so that its maximal biomass density lies at half the community power law.
`rfac`	A number that determines the strength of the non-linearity in the Beverton-Holt stock-recruitment relationship. The maximal recruitment will be set to rfac times the normal steady-state recruitment. Default value is 10.
`effort`	Default value is 0.

Value

An object of type MizerParams

Examples

```
## Not run:
params <- set_scaling_model(max_w_inf = 5000)
params <- setBackground(params)
a_m <- 0.0085
b_m <- 3.11
L_inf_m <- 24.3
L_mat <- 11.1
species_params <- data.frame(
    species = "mullet",
    w_min = 0.001,
    w_inf = a_m*L_inf_m^b_m,
    w_mat = a_m*L_mat^b_m,
    beta = 283,
    sigma = 1.8,
    z0 = 0,
    alpha = 0.6,
    sel_func = "knife_edge",
    knife_edge_size = 100,
    gear = "knife_edge_gear",
    k = 0,
    k_vb = 0.6,
    a = a_m,
    b = b_m
)
params <- addSpecies(params, species_params)
plotSpectra(params)
sim <- project(params, t_max=50)
plotBiomass(sim)

## End(Not run)
```

`display_frames` *Display frames*

Description

Display frames

Usage

```
display_frames(f1, f2, params, y_ticks = 6)
```

Arguments

f1	Data frame for left plot
f2	Data frame for right plot
params	A MizerParams object
y_ticks	The approximate number of ticks desired on the y axis

Value

ggplot2 object

`getBiomass` *Calculate the total biomass of each species within a size range at each time step.*

Description

Calculates the total biomass through time of the species in the `MizerSim` class within user defined size limits. The default option is to use the whole size range. You can specify minimum and maximum weight or length range for the species. Lengths take precedence over weights (i.e. if both min_l and min_w are supplied, only min_l will be used).

Usage

```
getBiomass(object, ...)

## S4 method for signature 'MizerSim'
getBiomass(object, ...)
```

Arguments

`object`	An object of class `MizerSim`.
`...`	Other arguments to select the size range of fish to be used in the calculation (min_w, max_w, min_l, max_l).

Value

An array containing the biomass (time x species)

Examples

```
## Not run:
data(NS_species_params_gears)
data(inter)
params <- MizerParams(NS_species_params_gears, inter)
# With constant fishing effort for all gears for 20 time steps
sim <- project(params, t_max = 20, effort = 0.5)
getBiomass(sim)
getBiomass(sim, min_w = 10, max_w = 1000)

## End(Not run)
```

getBiomassFrame *Get data frame of biomass of species through time, ready for ggplot2*

Description

After running a projection, the biomass of each species can be plotted against time. The biomass is calculated within user defined size limits (min_w, max_w, min_l, max_l, see `getBiomass`).

Usage

```
getBiomassFrame(sim, ...)

## S4 method for signature 'MizerSim'
getBiomassFrame(sim,
  species = sim@params@species_params$species[!is.na(sim@params@A)],
  start_time = as.numeric(dimnames(sim@n)[[1]][1]),
  end_time = as.numeric(dimnames(sim@n)[[1]][dim(sim@n)[1]]),
  ylim = c(NA, NA), total = FALSE, ...)
```

Arguments

`sim`	An object of class MizerSim
`...`	Other arguments to pass to `getBiomass` method, for example `min_w` and `max_w`
`species`	Name or vector of names of the species to be plotted. By default all species are plotted.
`start_time`	The first time to be plotted. Default is the beginning of the time series.
`end_time`	The last time to be plotted. Default is the end of the time series.
`ylim`	A numeric vector of length two providing limits of for the y axis. Use NA to refer to the existing minimum or maximum. Any values below 1e-20 are always cut off.
`total`	A boolean value that determines whether the total biomass from all species is plotted as well. Default is FALSE

Value

A data frame that can be used in `display_frames`

See Also

`getBiomass`

`getCommunitySlope` *Calculate the slope of the community abundance*

Description

Calculates the slope of the community abundance through time by performing a linear regression on the logged total numerical abundance at weight and logged weights (natural logs, not log to base 10, are used). You can specify minimum and maximum weight or length range for the species. Lengths take precedence over weights (i.e. if both min_l and min_w are supplied, only min_l will be used). You can also specify the species to be used in the calculation.

Usage

```
getCommunitySlope(object, ...)

## S4 method for signature 'MizerSim'
getCommunitySlope(object,
  species = 1:nrow(object@params@species_params), biomass = TRUE, ...)
```

Arguments

`object`	An object of class `MizerSim`.
`...`	Optional parameters include • min_w Minimum weight of species to be used in the calculation. • max_w Maximum weight of species to be used in the calculation. • min_l Minimum length of species to be used in the calculation. • max_l Maximum length of species to be used in the calculation.
`species`	Numeric or character vector of species to include in the calculation.
`biomass`	Boolean. If TRUE (default), the abundance is based on biomass, if FALSE the abundance is based on numbers.

Value

A data frame with slope, intercept and R2 values.

Examples

```
## Not run:
data(NS_species_params_gears)
data(inter)
params <- MizerParams(NS_species_params_gears, inter)
sim <- project(params, effort=1, t_max=40, dt = 1, t_save = 1)
# Slope based on biomass, using all species and sizes
slope_biomass <- getCommunitySlope(sim)
# Slope based on numbers, using all species and sizes
slope_numbers <- getCommunitySlope(sim, biomass=FALSE)
```

```
# Slope based on biomass, using all species and sizes between 10g and 1000g
slope_biomass <- getCommunitySlope(sim, min_w = 10, max_w = 1000)
# Slope based on biomass, using only demersal species and sizes between 10g and 1000g
dem_species <- c("Dab","Whiting","Sole","Gurnard","Plaice","Haddock", "Cod","Saithe")
slope_biomass <- getCommunitySlope(sim, species = dem_species, min_w = 10, max_w = 1000)

## End(Not run)
```

---

getEGrowth *Get energy rate available for growth*

Description

Calculates the energy rate $g_i(w)$ available by species and size for growth after metabolism, movement and reproduction have been accounted for. Used by the `project` method for performing simulations.

Usage

```
getEGrowth(object, n, n_pp, e_spawning, e)

## S4 method for signature 'MizerParams,matrix,numeric,matrix,matrix'
getEGrowth(object, n,
  n_pp, e_spawning, e)

## S4 method for signature 'MizerParams,matrix,numeric,missing,missing'
getEGrowth(object,
  n, n_pp)
```

Arguments

`object`	A MizerParams object.
`n`	A matrix of species abundance (species x size).
`n_pp`	A vector of the background abundance by size.
`e_spawning`	The energy available for spawning (optional, although if specified, e must also be specified). A matrix of size no. species x no. size bins. If not supplied, is calculated internally using the `getESpawning` method.
`e`	The energy available for reproduction and growth (optional, although if specified, e_spawning must also be specified). A matrix of size no. species x no. size bins. If not supplied, is calculated internally using the `getEReproAndGrowth` method.

Value

A two dimensional array (prey species x prey size)

See Also

`project`

Examples

```
## Not run:
data(NS_species_params_gears)
data(inter)
params <- MizerParams(NS_species_params_gears, inter)
# Project with constant fishing effort for all gears for 20 time steps
sim <- project(params, t_max = 20, effort = 0.5)
# Get the energy at a particular time step
getEGrowth(params,sim@n[21,,],sim@n_pp[21,])

## End(Not run)
```

`getEReproAndGrowth` *Get energy after metabolism and movement*

Description

Calculates the energy rate available by species and size for reproduction and growth after metabolism and movement have been accounted for: $E_{r.i}(w)$. Used by the `project` method for performing simulations.

Usage

```
getEReproAndGrowth(object, n, n_pp, feeding_level)

## S4 method for signature 'MizerParams,matrix,numeric,matrix'
getEReproAndGrowth(object, n,
  n_pp, feeding_level)

## S4 method for signature 'MizerParams,matrix,numeric,missing'
getEReproAndGrowth(object,
  n, n_pp)
```

Arguments

`object`	A `MizerParams` object.
`n`	A matrix of species abundance (species x size).
`n_pp`	A vector of the background abundance by size.
`feeding_level`	The current feeding level (optional). A matrix of size no. species x no. size bins. If not supplied, is calculated internally using the `getFeedingLevel()` method.

Value

A two dimensional array (species x size)

See Also

project and getFeedingLevel.

Examples

```
## Not run:
data(NS_species_params_gears)
data(inter)
params <- MizerParams(NS_species_params_gears, inter)
# Project with constant fishing effort for all gears for 20 time steps
sim <- project(params, t_max = 20, effort = 0.5)
# Get the energy at a particular time step
getEReproAndGrowth(params,sim@n[21,,],sim@n_pp[21,])

## End(Not run)
```

---

getESpawning — *Get energy rate available for reproduction*

Description

Calculates the energy rate available by species and size for reproduction after metabolism and movement have been accounted for: $\psi_i(w)E_{r.i}(w)$. Used by the `project` method for performing simulations.

Usage

```
getESpawning(object, n, n_pp, e)

## S4 method for signature 'MizerParams,matrix,numeric,matrix'
getESpawning(object, n, n_pp,
  e)

## S4 method for signature 'MizerParams,matrix,numeric,missing'
getESpawning(object, n,
  n_pp)
```

Arguments

object	A `MizerParams` object.
n	A matrix of species abundance (species x size).
n_pp	A vector of the background abundance by size.
e	The energy available for reproduction and growth (optional). A matrix of size no. species x no. size bins. If not supplied, is calculated internally using the `getEReproAndGrowth()` method.

Value

A two dimensional array (prey species x prey size)

See Also

project and getEReproAndGrowth.

Examples

```
## Not run:
data(NS_species_params_gears)
data(inter)
params <- MizerParams(NS_species_params_gears, inter)
# Project with constant fishing effort for all gears for 20 time steps
sim <- project(params, t_max = 20, effort = 0.5)
# Get the energy at a particular time step
getESpawning(params,sim@n[21,,],sim@n_pp[21,])

## End(Not run)
```

---

getFeedingLevel *Get feeding level*

Description

Calculates the feeding level $f_i(w)$ as a by predator size based on food availability, search volume and maximum intake. The feeding level is the proportion of the encountered food that is actually consumed.This method is used by the project method for performing simulations.

Usage

```
getFeedingLevel(object, n, n_pp, phi_prey, ...)

## S4 method for signature 'MizerParams,matrix,numeric,matrix'
getFeedingLevel(object, n,
  n_pp, phi_prey, ...)

## S4 method for signature 'MizerParams,matrix,numeric,missing'
getFeedingLevel(object, n,
  n_pp, phi_prey, ...)

## S4 method for signature 'MizerSim,missing,missing,missing'
getFeedingLevel(object,
  time_range = dimnames(object@n)$time, drop = FALSE, ...)
```

Arguments

`object`	A `MizerParams` or `MizerSim` object
`n`	A matrix of species abundance (species x size). Only used if `object` argument is of type `MizerParams`.
`n_pp`	A vector of the background abundance by size. Only used if `object` argument is of type `MizerParams`.
`phi_prey`	The PhiPrey matrix (optional) of dimension no. species x no. size bins. If not passed in, it is calculated internally using the `getPhiPrey` method. Only used if `object` argument is of type `MizerParams`.
`...`	Other arguments (currently unused).
`time_range`	Subset the returned fishing mortalities by time. The time range is either a vector of values, a vector of min and max time, or a single value. Default is the whole time range. Only used if the `object` argument is of type `MizerSim`.
`drop`	should extra dimensions of length 1 in the output be dropped, simplifying the output. Defaults to TRUE.

Note

If a `MizerParams` object is passed in, the method returns a two dimensional array (predator species x predator size) based on the abundances also passed in.

If a `MizerSim` object is passed in, the method returns a three dimensional array (time step x predator species x predator size) with the feeding level calculated at every time step in the simulation.

See Also

`getPhiPrey`

Examples

```
## Not run:
data(NS_species_params_gears)
data(inter)
params <- MizerParams(NS_species_params_gears, inter)
# With constant fishing effort for all gears for 20 time steps
sim <- project(params, t_max = 20, effort = 0.5)
# Get the feeding level at one time step
n <- sim@n[21,,]
n_pp <- sim@n_pp[21,]
fl <- getFeedingLevel(params,n,n_pp)
# Get the feeding level at all saved time steps
fl <- getFeedingLevel(sim)
# Get the feeding level for time 15 - 20
fl <- getFeedingLevel(sim, time_range = c(15,20))

## End(Not run)
```

`getFMort` *Get the total fishing mortality rate from all fishing gears by time, species and size.*

Description

Calculates the total fishing mortality from all gears by species and size at each time step in the `effort` argument. The total fishing mortality is just the sum of the fishing mortalities imposed by each gear, $\mu_{f.i}(w) = \sum_g F_{g,i,w}$.

Usage

```
getFMort(object, effort, ...)

## S4 method for signature 'MizerParams,numeric'
getFMort(object, effort, ...)

## S4 method for signature 'MizerParams,matrix'
getFMort(object, effort, ...)

## S4 method for signature 'MizerSim,missing'
getFMort(object, effort,
  time_range = dimnames(object@effort)$time, drop = TRUE, ...)
```

Arguments

`object`	A `MizerParams` object or a `MizerSim` object
`effort`	The effort of each fishing gear. Only needed if the object argument is of class `MizerParams`. See notes below.
`...`	Other arguments passed to `getFMortGear` method.
`time_range`	Subset the returned fishing mortalities by time. The time range is either a vector of values, a vector of min and max time, or a single value. Default is the whole time range. Only used if the `object` argument is of type `MizerSim`.
`drop`	Only used when object is of type `MizerSim`. Should dimensions of length 1 be dropped, e.g. if your community only has one species it might make presentation of results easier. Default is TRUE

Value

An array. If the effort argument has a time dimension, or object is of class `MizerSim`, the output array has three dimensions (time x species x size). If the effort argument does not have a time dimension, the output array has two dimensions (species x size).

Note

Here: fishing mortality = catchability x selectivity x effort.

The `effort` argument is only used if a `MizerParams` object is passed in. The `effort` argument can be a two dimensional array (time x gear), a vector of length equal to the number of gears (each gear has a different effort that is constant in time), or a single numeric value (each gear has the same effort that is constant in time). The order of gears in the `effort` argument must be the same the same as in the `MizerParams` object.

If the object argument is of class `MizerSim` then the effort slot of the `MizerSim` object is used and the `effort` argument is not used.

See Also

`getFMortGear`, `project`

Examples

```
## Not run:
data(NS_species_params_gears)
data(inter)
params <- MizerParams(NS_species_params_gears, inter)
# Get the total fishing mortality when effort is constant for all
# gears and time:
getFMort(params, effort = 1)
# Get the total fishing mortality when effort is different
# between the four gears but constant in time:
getFMort(params, effort = c(0.5,1,1.5,0.75))
# Get the total fishing mortality when effort is different
# between the four gears and changes with time:
effort <- array(NA, dim = c(20,4))
effort[,1] <- seq(from=0, to = 1, length=20)
effort[,2] <- seq(from=1, to = 0.5, length=20)
effort[,3] <- seq(from=1, to = 2, length=20)
effort[,4] <- seq(from=2, to = 1, length=20)
getFMort(params, effort=effort)
# Get the total fishing mortality using the effort already held in a
# MizerSim object.
sim <- project(params, t_max = 20, effort = 0.5)
getFMort(sim)
getFMort(sim, time_range = c(10,20))

## End(Not run)
```

---

getFMortGear *Get the fishing mortality by time, gear, species and size*

Description

Calculates the fishing mortality rate $F_{g,i,w}$ by gear, species and size at each time step in the `effort` argument. Used by the `project` method to perform simulations.

Usage

```
getFMortGear(object, effort, ...)

## S4 method for signature 'MizerParams,numeric'
getFMortGear(object, effort, ...)

## S4 method for signature 'MizerParams,matrix'
getFMortGear(object, effort, ...)

## S4 method for signature 'MizerSim,missing'
getFMortGear(object, effort,
  time_range = dimnames(object@effort)$time, ...)
```

Arguments

`object`	A `MizerParams` object or a `MizerSim` object.
`effort`	The effort of each fishing gear. Only needed if the object argument is of class `MizerParams`. See notes below.
`...`	Other arguments (currently unused).
`time_range`	Subset the returned fishing mortalities by time. The time range is either a vector of values, a vector of min and max time, or a single value. Default is the whole time range. Only used if the `object` argument is of type `MizerSim`.

Value

An array. If the effort argument has a time dimension, or a `MizerSim` is passed in, the output array has four dimensions (time x gear x species x size). If the effort argument does not have a time dimension (i.e. it is a vector or a single numeric), the output array has three dimensions (gear x species x size).

Note

Here: fishing mortality = catchability x selectivity x effort.

The `effort` argument is only used if a `MizerParams` object is passed in. The `effort` argument can be a two dimensional array (time x gear), a vector of length equal to the number of gears (each gear has a different effort that is constant in time), or a single numeric value (each gear has the same effort that is constant in time). The order of gears in the `effort` argument must be the same the same as in the `MizerParams` object.

If the object argument is of class `MizerSim` then the effort slot of the `MizerSim` object is used and the `effort` argument is not used.

Examples

```
## Not run:
data(NS_species_params_gears)
data(inter)
params <- MizerParams(NS_species_params_gears, inter)
# Get the fishing mortality when effort is constant
# for all gears and time:
getFMortGear(params, effort = 1)
# Get the fishing mortality when effort is different
# between the four gears but constant in time:
getFMortGear(params, effort = c(0.5,1,1.5,0.75))
# Get the fishing mortality when effort is different
# between the four gears and changes with time:
effort <- array(NA, dim = c(20,4))
effort[,1] <- seq(from=0, to = 1, length=20)
effort[,2] <- seq(from=1, to = 0.5, length=20)
effort[,3] <- seq(from=1, to = 2, length=20)
effort[,4] <- seq(from=2, to = 1, length=20)
getFMortGear(params, effort=effort)
# Get the fishing mortality using the effort already held in a MizerSim object.
sim <- project(params, t_max = 20, effort = 0.5)
getFMortGear(sim)
getFMortGear(sim, time_range=c(10,20))

## End(Not run)
```

getM2 *getM2 method for the size based model*

Description

Calculates the total predation mortality rate $\mu_{p,i}(w_p)$ on each prey species by prey size. This method is used by the `project` method for performing simulations.

Usage

```
getM2(object, n, n_pp, pred_rate, ...)

## S4 method for signature 'MizerParams,missing,missing,array'
getM2(object, pred_rate)

## S4 method for signature 'MizerParams,matrix,numeric,missing'
getM2(object, n, n_pp)

## S4 method for signature 'MizerSim,missing,missing,missing'
getM2(object,
  time_range = dimnames(object@n)$time, drop = TRUE, ...)
```

Arguments

`object`	A `MizerParams` or `MizerSim` object.
`n`	A matrix of species abundance (species x size). Only used if `object` argument is of type `MizerParams`.

`n_pp`	A vector of the background abundance by size. Only used if `object` argument is of type `MizerParams`.
`pred_rate`	An array of predation rates of dimension no. sp x no. community size bins x no. of size bins in whole spectra (i.e. community + background, the w_full slot). The array is optional. If it is not provided it is calculated by the `getPredRate()` method.
`...`	Other arguments (currently unused).
`time_range`	Subset the returned fishing mortalities by time. The time range is either a vector of values, a vector of min and max time, or a single value. Default is the whole time range. Only used if the `object` argument is of type `MizerSim`.
`drop`	Only used when object is of type `MizerSim`. Should dimensions of length 1 in the output be dropped, simplifying the output. Defaults to TRUE

Value

If a `MizerParams` object is passed in, the method returns a two dimensional array (prey species x prey size) based on the abundances also passed in. If a `MizerSim` object is passed in, the method returns a three dimensional array (time step x prey species x prey size) with the predation mortality calculated at every time step in the simulation.

See Also

`getPredRate` and `project`.

Examples

```
## Not run:
data(NS_species_params_gears)
data(inter)
params <- MizerParams(NS_species_params_gears, inter)
# With constant fishing effort for all gears for 20 time steps
sim <- project(params, t_max = 20, effort = 0.5)
# Get M2 at one time step
n <- sim@n[21,,]
n_pp <- sim@n_pp[21,]
getM2(params,n,n_pp)
# Get M2 at all saved time steps
getM2(sim)
# Get M2 over the time 15 - 20
getM2(sim, time_range = c(15,20))

## End(Not run)
```

getM2Background *Get predation mortality rate for plankton*

Description

Calculates the predation mortality rate $\mu_p(w)$ on the plankton spectrum by plankton size. Used by the `project` method for running size based simulations.

Usage

```
getM2Background(object, n, n_pp, pred_rate)

## S4 method for signature 'MizerParams,matrix,numeric,array'
getM2Background(object, n,
  n_pp, pred_rate)

## S4 method for signature 'MizerParams,matrix,numeric,missing'
getM2Background(object, n,
  n_pp, pred_rate)
```

Arguments

`object`	A `MizerParams` object.
`n`	A matrix of species abundance (species x size).
`n_pp`	A vector of the background abundance by size.
`pred_rate`	An array of predation rates of dimension no. sp x no. community size bins x no. of size bins in whole spectra (i.e. community + background, the w_full slot). The array is optional. If it is not provided it is calculated by the `getPredRate()` method.

Value

A vector of predation mortalities by background prey size.

See Also

`project` and `getM2`.

Examples

```
## Not run:
data(NS_species_params_gears)
data(inter)
params <- MizerParams(NS_species_params_gears, inter)
# With constant fishing effort for all gears for 20 time steps
sim <- project(params, t_max = 20, effort = 0.5)
# Get M2 of the background spectrum at one time step
```

```
n <- sim@n[21,,]
n_pp <- sim@n_pp[21,]
getM2Background(params,n,n_pp)

## End(Not run)
```

getMeanMaxWeight *Calculate the mean maximum weight of the community*

Description

Calculates the mean maximum weight of the community through time. This can be calculated by numbers or biomass. The calculation is the sum of the w_inf * abundance of each species, divided by the total abundance community, where abundance is either in biomass or numbers. You can specify minimum and maximum weight or length range for the species. Lengths take precedence over weights (i.e. if both min_l and min_w are supplied, only min_l will be used). You can also specify the species to be used in the calculation.

Usage

```
getMeanMaxWeight(object, ...)

## S4 method for signature 'MizerSim'
getMeanMaxWeight(object,
  species = 1:nrow(object@params@species_params), measure = "both",
  ...)
```

Arguments

`object`	An object of class `MizerSim`.
`...`	Other arguments for the `getN` and `getBiomass` methods such as `min_w`, `max_w` `min_l` and `max_l`.
`species`	numeric or character vector of species to include in the calculation.
`measure`	The measure to return. Can be 'numbers', 'biomass' or 'both'

Value

A matrix or vector containing the mean maximum weight of the community through time

Examples

```
## Not run:
data(NS_species_params_gears)
data(inter)
params <- MizerParams(NS_species_params_gears, inter)
sim <- project(params, effort=1, t_max=10)
getMeanMaxWeight(sim)
getMeanMaxWeight(sim, species=c("Herring","Sprat","N.pout"))
getMeanMaxWeight(sim, min_w = 10, max_w = 5000)

## End(Not run)
```

`getMeanWeight` *Calculate the mean weight of the community*

Description

Calculates the mean weight of the community through time. This is simply the total biomass of the community divided by the abundance in numbers. You can specify minimum and maximum weight or length range for the species. Lengths take precedence over weights (i.e. if both min_l and min_w are supplied, only min_l will be used). You can also specify the species to be used in the calculation.

Usage

```
getMeanWeight(object, ...)

## S4 method for signature 'MizerSim'
getMeanWeight(object,
  species = 1:nrow(object@params@species_params), ...)
```

Arguments

`object`	An object of class `MizerSim`
`...`	Other arguments for the `getN` and `getBiomass` methods such as `min_w`, `max_w` `min_l` and `max_l`.
`species`	numeric or character vector of species to include in the calculation

Value

A vector containing the mean weight of the community through time

Examples

```
## Not run:
data(NS_species_params_gears)
data(inter)
params <- MizerParams(NS_species_params_gears, inter)
sim <- project(params, effort=1, t_max=10)
getMeanWeight(sim)
getMeanWeight(sim, species=c("Herring","Sprat","N.pout"))
getMeanWeight(sim, min_w = 10, max_w = 5000)

## End(Not run)
```

getN — *Calculate the total abundance in terms of numbers of species within a size range*

Description

Calculates the total numbers through time of the species in the `MizerSim` class within user defined size limits. The default option is to use the whole size range You can specify minimum and maximum weight or lengths for the species. Lengths take precedence over weights (i.e. if both min_l and min_w are supplied, only min_l will be used)

Usage

```
getN(object, ...)

## S4 method for signature 'MizerSim'
getN(object, ...)
```

Arguments

`object`	An object of class `MizerSim`.
`...`	Other arguments to select the size range of the species to be used in the calculation (min_w, max_w, min_l, max_l).

Value

An array containing the total numbers (time x species)

Examples

```
## Not run:
data(NS_species_params_gears)
data(inter)
params <- MizerParams(NS_species_params_gears, inter)
# With constant fishing effort for all gears for 20 time steps
sim <- project(params, t_max = 20, effort = 0.5)
getN(sim)
getN(sim, min_w = 10, max_w = 1000)

## End(Not run)
```

getPhiPrey *Get available energy*

Description

Calculates the amount $E_{a,i}(w)$ of food exposed to each predator as a function of predator size.

Usage

```
getPhiPrey(object, n, n_pp, ...)

## S4 method for signature 'MizerParams,matrix,numeric'
getPhiPrey(object, n, n_pp, ...)
```

Arguments

`object`	An MizerParams object
`n`	A matrix of species abundances (species x size)
`n_pp`	A vector of the background abundance by size
`...`	Other arguments (currently unused)

Details

This method is used by the `project` method for performing simulations.

Value

A two dimensional array (predator species x predator size)

See Also

`project`

Examples

```
## Not run:
data(NS_species_params_gears)
data(inter)
params <- MizerParams(NS_species_params_gears, inter)
# With constant fishing effort for all gears for 20 time steps
sim <- project(params, t_max = 20, effort = 0.5)
n <- sim@n[21,,]
n_pp <- sim@n_pp[21,]
getPhiPrey(params,n,n_pp)

## End(Not run)
```

getPredRate *Get predation rate*

Description

Calculates the predation rate of each predator species at size on prey size. In formulas

$$\int \phi_i(w_p/w)(1-f_i(w))\gamma_i w^q N_i(w) dw$$

This method is used by the project method for performing simulations. In the simulations, it is combined with the interaction matrix (see MizerParams) to calculate the realised predation mortality (see getM2).

Usage

```
getPredRate(object, n, n_pp, feeding_level)

## S4 method for signature 'MizerParams,matrix,numeric,matrix'
getPredRate(object, n, n_pp,
  feeding_level)

## S4 method for signature 'MizerParams,matrix,numeric,missing'
getPredRate(object, n, n_pp)
```

Arguments

`object`	A `MizerParams` object.
`n`	A matrix of species abundance (species x size).
`n_pp`	A vector of the background abundance by size.
`feeding_level`	The current feeding level (optional). A matrix of size no. species x no. size bins. If not supplied, is calculated internally using the `getFeedingLevel()` method.

Value

A two dimensional array (predator species x prey size), where the prey size runs over community plus background spectrum.

See Also

project, getM2, getFeedingLevel and MizerParams

Examples

```
## Not run:
data(NS_species_params_gears)
data(inter)
params <- MizerParams(NS_species_params_gears, inter)
# With constant fishing effort for all gears for 20 time steps
sim <- project(params, t_max = 20, effort = 0.5)
# Get the feeding level at one time step
n <- sim@n[21,,]
n_pp <- sim@n_pp[21,]
getPredRate(params,n,n_pp)

## End(Not run)
```

---

getProportionOfLargeFish

Calculate the proportion of large fish

Description

Calculates the proportion of large fish through time in the `MizerSim` class within user defined size limits. The default option is to use the whole size range. You can specify minimum and maximum size ranges for the species and also the threshold size for large fish. Sizes can be expressed as weight or size. Lengths take precedence over weights (i.e. if both min_l and min_w are supplied, only min_l will be used). You can also specify the species to be used in the calculation. This method can be used to calculate the Large Fish Index. The proportion is based on either abundance or biomass.

Usage

```
getProportionOfLargeFish(object, ...)

## S4 method for signature 'MizerSim'
getProportionOfLargeFish(object,
  species = 1:nrow(object@params@species_params), threshold_w = 100,
  threshold_l = NULL, biomass_proportion = TRUE, ...)
```

Arguments

`object`	An object of class `MizerSim`.
`...`	Other arguments to select the size range of the species to be used in the calculation (min_w, max_w, min_l, max_l).
`species`	numeric or character vector of species to include in the calculation.
`threshold_w`	the size used as the cutoff between large and small fish. Default value is 100.
`threshold_l`	the size used as the cutoff between large and small fish.
`biomass_proportion`	a boolean value. If TRUE the proportion calculated is based on biomass, if FALSE it is based on numbers of individuals. Default is TRUE.

Value

An array containing the proportion of large fish through time

Examples

```
## Not run:
data(NS_species_params_gears)
data(inter)
params <- MizerParams(NS_species_params_gears, inter)
sim <- project(params, effort=1, t_max=10)
getProportionOfLargeFish(sim)
getProportionOfLargeFish(sim, species=c("Herring","Sprat","N.pout"))
getProportionOfLargeFish(sim, min_w = 10, max_w = 5000)
getProportionOfLargeFish(sim, min_w = 10, max_w = 5000, threshold_w = 500)
getProportionOfLargeFish(sim, min_w = 10, max_w = 5000,
    threshold_w = 500, biomass_proportion=FALSE)

## End(Not run)
```

`getRDD` *Get density dependent recruitment*

Description

Calculates the density dependent recruitment (total egg production) R_i for each species. This is the flux entering the smallest size class of each species. The density dependent recruitment is the density independent recruitment after it has been put through the density dependent stock-recruitment relationship function. This method is used by the `project` method for performing simulations.

Usage

```
getRDD(object, n, n_pp, rdi, ...)

## S4 method for signature 'MizerParams,matrix,numeric,matrix'
getRDD(object, n, n_pp, rdi,
  sex_ratio = 0.5)

## S4 method for signature 'MizerParams,matrix,numeric,missing'
getRDD(object, n, n_pp,
  sex_ratio = 0.5)
```

Arguments

`object`	An `MizerParams` object
`n`	A matrix of species abundance (species x size)
`n_pp`	A vector of the background abundance by size

`rdi`	A matrix of density independent recruitment (optional) with dimensions no. sp x 1. If not specified rdi is calculated internally using the `getRDI` method.
`...`	Other arguments (currently unused).
`sex_ratio`	Proportion of the population that is female. Default value is 0.5

Value

A numeric vector the length of the number of species.

Examples

```
## Not run:
data(NS_species_params_gears)
data(inter)
params <- MizerParams(NS_species_params_gears, inter)
# Project with constant fishing effort for all gears for 20 time steps
sim <- project(params, t_max = 20, effort = 0.5)
# Get the energy at a particular time step
getRDD(params,sim@n[21,,],sim@n_pp[21,])

## End(Not run)
```

`getRDI` *Get density independent recruitment*

Description

Calculates the density independent recruitment (total egg production) $R_{p.i}$ before density dependence, by species. Used by the `project` method for performing simulations.

Usage

```
getRDI(object, n, n_pp, e_spawning, ...)

## S4 method for signature 'MizerParams,matrix,numeric,matrix'
getRDI(object, n, n_pp,
  e_spawning, sex_ratio = 0.5)

## S4 method for signature 'MizerParams,matrix,numeric,missing'
getRDI(object, n, n_pp,
  sex_ratio = 0.5)
```

Arguments

`object`	A `MizerParams` object.
`n`	A matrix of species abundance (species x size).
`n_pp`	A vector of the background abundance by size.

`e_spawning`	The energy available for spawning (optional). A matrix of size no. species x no. size bins. If not supplied, is calculated internally using the `getESpawning` method.
`...`	Other arguments (currently unused).
`sex_ratio`	Proportion of the population that is female. Default value is 0.5.

Value

A numeric vector the length of the number of species

See Also

`project`

Examples

```
## Not run:
data(NS_species_params_gears)
data(inter)
params <- MizerParams(NS_species_params_gears, inter)
# Project with constant fishing effort for all gears for 20 time steps
sim <- project(params, t_max = 20, effort = 0.5)
# Get the recruitment at a particular time step
getRDI(params,sim@n[21,,],sim@n_pp[21,])

## End(Not run)
```

---

getSSB *Calculate the SSB of species*

Description

Calculates the spawning stock biomass (SSB) through time of the species in the `MizerSim` class. SSB is calculated as the total mass of all mature individuals.

Usage

```
getSSB(object)

## S4 method for signature 'MizerSim'
getSSB(object)
```

Arguments

`object`	An object of class `MizerSim`.

Value

An array containing the SSB (time x species)

Examples

```
## Not run:
data(NS_species_params_gears)
data(inter)
params <- MizerParams(NS_species_params_gears, inter)
# With constant fishing effort for all gears for 20 time steps
sim <- project(params, t_max = 20, effort = 0.5)
getSSB(sim)

## End(Not run)
```

---

getSSBFrame — *Get data frame of spawning stock biomass of species through time, ready for ggplot2*

Description

After running a projection, the spawning stock biomass of each species can be plotted against time.

Usage

```
getSSBFrame(sim, ...)

## S4 method for signature 'MizerSim'
getSSBFrame(sim,
  species = sim@params@species_params$species[!is.na(sim@params@A)],
  start_time = as.numeric(dimnames(sim@n)[[1]][1]),
  end_time = as.numeric(dimnames(sim@n)[[1]][dim(sim@n)[1]]),
  ylim = c(NA, NA), total = FALSE, ...)
```

Arguments

`sim`	An object of class MizerSim
`...`	Other arguments to pass to `getSBB` method, currently unused.
`species`	Name or vector of names of the species to be plotted. By default all species are plotted.
`start_time`	The first time to be plotted. Default is the beginning of the time series.
`end_time`	The last time to be plotted. Default is the end of the time series.
`ylim`	A numeric vector of length two providing limits of for the y axis. Use NA to refer to the existing minimum or maximum. Any values below 1e-20 are always cut off.
`total`	A boolean value that determines whether the total SSB from all species is plotted as well. Default is FALSE

Value

A data frame that can be used in `display_frames`

See Also

getSSB

getYield — *Calculate the total yield of each species*

Description

Calculates the total yield of each species across all gears at each simulation time step.

Usage

```
getYield(object)

## S4 method for signature 'MizerSim'
getYield(object)
```

Arguments

object	An object of class `MizerSim`.

Value

An array containing the total yield (time x species)

See Also

getYieldGear

Examples

```
## Not run:
data(NS_species_params_gears)
data(inter)
params <- MizerParams(NS_species_params_gears, inter)
sim <- project(params, effort=1, t_max=10)
y <- getYield(sim)

## End(Not run)
```

getYieldGear *Calculate the total yield per gear and species*

Description

Calculates the total yield per gear and species at each simulation time step.

Usage

```
getYieldGear(object)

## S4 method for signature 'MizerSim'
getYieldGear(object)
```

Arguments

object	An object of class `MizerSim`.

Value

An array containing the total yield (time x gear x species)

See Also

getYield

Examples

```
## Not run:
data(NS_species_params_gears)
data(inter)
params <- MizerParams(NS_species_params_gears, inter)
# With constant fishing effort for all gears for 20 time steps
sim <- project(params, t_max = 20, effort = 0.5)
getYieldGear(sim)

## End(Not run)
```

getZ *Get total mortality rate*

Description

Calculates the total mortality rate $\mu_i(w)$ on each species by size from predation mortality, background mortality and fishing mortality for a single time step.

Usage

```
getZ(object, n, n_pp, effort, m2)

## S4 method for signature 'MizerParams,matrix,numeric,numeric,matrix'
getZ(object, n, n_pp,
  effort, m2)

## S4 method for signature 'MizerParams,matrix,numeric,numeric,missing'
getZ(object, n,
  n_pp, effort)
```

Arguments

`object`	A `MizerParams` object.
`n`	A matrix of species abundance (species x size).
`n_pp`	A vector of the background abundance by size.
`effort`	A numeric vector of the effort by gear or a single numeric effort value which is used for all gears.
`m2`	A two dimensional array of predation mortality (optional). Has dimensions no. sp x no. size bins in the community. If not supplied is calculated using the `getM2()` method.

Value

A two dimensional array (prey species x prey size).

See Also

`getM2`, `getFMort`

Examples

```
## Not run:
data(NS_species_params_gears)
data(inter)
params <- MizerParams(NS_species_params_gears, inter)
# Project with constant fishing effort for all gears for 20 time steps
sim <- project(params, t_max = 20, effort = 0.5)
# Get the total mortality at a particular time step
getZ(params,sim@n[21,,],sim@n_pp[21,],effort=0.5)

## End(Not run)
```

get_initial_n — *Calculate initial population abundances for the community populations*

Description

This function uses the model parameters and other parameters to calculate initial population abundances for the community populations. These initial abundances should be reasonable guesses at the equilibrium values. The returned population can be passed to the `project` method.

Usage

```
get_initial_n(params, n0_mult = NULL, a = 0.35)
```

Arguments

`params`	The model parameters. An object of type MizerParams.
`n0_mult`	Multiplier for the abundance at size 0. Default value is kappa/1000.
`a`	A parameter with a default value of 0.35.

Value

A matrix (species x size) of population abundances.

Examples

```
## Not run:
data(NS_species_params_gears)
params <- MizerParams(NS_species_params_gears)
init_n <- get_initial_n(params)

## End(Not run)
```

inter — *Example interaction matrix for the North Sea example*

Description

The interaction coefficient between predators and preys in the North Sea.

Format

A 12 x 12 matrix.

Source

Blanchard et al.

`knife_edge` *Size based knife-edge selectivity function*

Description

A knife-edge selectivity function where only sizes greater or equal to `knife_edge_size` are selected.

Usage

```
knife_edge(w, knife_edge_size)
```

Arguments

`w`	The size of the individual.
`knife_edge_size`	The size at which the knife-edge operates.

`log_breaks` *Helper function to produce nice breaks on logarithmic axes*

Description

This is needed when the logarithmic y-axis spans less than one order of magnitude, in which case the ggplot2 default produces no ticks. Thanks to Heather Turner at https://stackoverflow.com/questions/14255533/pretty-ticks-for-log-normal-scale-using-ggplot2-dynamic-not-manual

Usage

```
log_breaks(n = 6)
```

Arguments

`n`	Approximate number of ticks

Value

A function that can be used as the break argument in calls to scale_y_continuous() or scale_x_continuous()

`mizer` *mizer: Multi-species size-based modelling in R*

Description

The mizer package implements multi-species size-based modelling in R. It has been designed for modelling marine ecosystems.

Details

Using **mizer** is relatively simple. There are three main stages:

1. Setting the model parameters. This is done by creating an object of class `MizerParams`. This includes model parameters such as the life history parameters of each species, and the range of the size spectrum.
2. Running a simulation. This is done by calling the `project()` method on the model parameters. This produces an object of `MizerSim` which contains the results of the simulation.
3. Exploring results. After a simulation has been run, the results can be explored using a range of plots and summaries.

See the accompanying vignette for full details of the principles behind mizer and how the package can be used to perform size-based modelling.

`MizerParams` *Constructors for objects of* `MizerParams` *class*

Description

Constructor method for the MizerParams class. Provides the simplest way of making a `MizerParams` object to be used in a simulation.

Only really used to make MizerParams of the right size and shouldn't be used by user

Usage

```
MizerParams(object, interaction, ...)

## S4 method for signature 'numeric,missing'
MizerParams(object, min_w = 0.001,
  max_w = 1000, no_w = 100, min_w_pp = 1e-10, no_w_pp = NA,
  species_names = 1:object, gear_names = species_names)

## S4 method for signature 'data.frame,matrix'
MizerParams(object, interaction, n = 2/3,
  p = 0.7, q = 0.8, r_pp = 10, kappa = 1e+11, lambda = (2 + q -
  n), w_pp_cutoff = 10, max_w = max(object$w_inf) * 1.1, f0 = 0.6,
  z0pre = 0.6, z0exp = n - 1, ...)

## S4 method for signature 'data.frame,missing'
MizerParams(object, interaction, ...)
```

Arguments

`object`	A data frame of species specific parameter values (see notes below).
`interaction`	Optional argument to specify the interaction matrix of the species (predator by prey). If missing a default interaction is used where all interactions between species are set to 1. Note that any dimnames of the interaction matrix argument are ignored by the constructor. The dimnames of the interaction matrix in the returned `MizerParams` object are taken from the species names in the `species_params` slot. This means that the order of the columns and rows of the interaction matrix argument should be the same as the species name in the `species_params` slot.
`...`	Additional arguments.
`min_w`	The smallest size of the community spectrum.
`max_w`	The largest size of the community spectrum. Default value is the largest w_inf in the community x 1.1.
`no_w`	The number of size bins in the community spectrum.
`min_w_pp`	The smallest size of the background spectrum.
`no_w_pp`	Obsolete argument that is no longer used because the number of plankton size bins is determined because all size bins have to be logarithmically equally spaced.
`species_names`	Names of the species. Generally not needed as normally taken from the `object` data.frame.
`gear_names`	Names of the gears that catch each species. Generally not needed as normally taken from the `object` data.frame. Default is `species_names`.
`n`	Scaling of the intake. Default value is 2/3.
`p`	Scaling of the standard metabolism. Default value is 0.7.
`q`	Exponent of the search volume. Default value is 0.8.
`r_pp`	Growth rate of the primary productivity. Default value is 10.
`kappa`	Carrying capacity of the resource spectrum. Default value is 1e11.
`lambda`	Exponent of the resource spectrum. Default value is (2+q-n).
`w_pp_cutoff`	The cut off size of the background spectrum. Default value is 10.
`f0`	Average feeding level. Used to calculated `h` and `gamma` if those are not columns in the species data frame. Also requires `k_vb` (the von Bertalanffy K parameter) to be a column in the species data frame. If `h` and `gamma` are supplied then this argument is ignored. Default is 0.6..
`z0pre`	If `z0`, the mortality from other sources, is not a column in the species data frame, it is calculated as z0pre * w_inf ^ z0exp. Default value is 0.6.
`z0exp`	If `z0`, the mortality from other sources, is not a column in the species data frame, it is calculated as z0pre * w_inf ^ z0exp. Default value is n-1.

Value

An object of type `MizerParams`

Note

The only essential argument to the `MizerParams` constructor is a data frame which contains the species data. The data frame is arranged species by parameter, so each column of the parameter data frame is a parameter and each row has the parameters for one of the species in the model.

There are some essential columns that must be included in the parameter data.frame and that do not have default values. Other columns do have default values, so that if they are not included in the species parameter data frame, they will be automatically added when the `MizerParams` object is created. See the accompanying vignette for details of these columns.

An additional constructor method which takes an integer of the number of species in the model. This is only used in internally to set up a `MizerParams` object with the correct dimensions. It is not recommended that this method is used by users.

See Also

`project` MizerSim

Examples

```
data(NS_species_params_gears)
data(inter)
params <- MizerParams(NS_species_params_gears, inter)
```

`MizerParams-class` *A class to hold the parameters for a size based model.*

Description

These parameters include the model species, their life history parameters and the size ranges of the model.

Details

MizerParams objects can be created using a range of `MizerParams` constructor methods.

Dynamic simulations are performed using the `project` method on objects of this class.

Slots

`w` A numeric vector of size bins used for the community (i.e. fish) part of the model. These are usually spaced on a log10 scale

`dw` The absolute difference between the size bins specified in the w slot. A vector the same length as the w slot. The final value is the same as the second to last value `w_full` A numeric vector of size bins used for the whole model (i.e. the community and background spectra) . These are usually spaced on a log10 scale

`dw_full` The absolute difference between the size bins specified in the w_full slot. A vector the same length as the w_full slot. The final value is the same as the second to last value

`psi` An array (species x size) that holds the allocation to reproduction for each species at size, $\psi_i(w)$

`intake_max` An array (species x size) that holds the maximum intake for each species at size, $h_i w^n$

`search_vol` An array (species x size) that holds the search volume for each species at size, $\gamma_i w^q$

`activity` An array (species x size) that holds the activity for each species at size, $k_i w$

`std_metab` An array (species x size) that holds the standard metabolism for each species at size, $k_{s.i} w^p$

`mu_b` An array (species x size) that holds the background death $\mu_{b.i}(w)$

`ft_pred_kernel_e` An array (species x log of predator/prey size ratio) that holds the Fourier transform of the feeding kernel in a form appropriate for evaluating the available energy integral

`ft_pred_kernel_p` An array (species x log of predator/prey size ratio) that holds the Fourier transform of the feeding kernel in a form appropriate for evaluating the predation mortality integral

`rr_pp` A vector the same length as the w_full slot. The size specific growth rate of the background spectrum, $r_0 w^{p-1}$

`cc_pp` A vector the same length as the w_full slot. The size specific carrying capacity of the background spectrum, $\kappa w^{-\lambda}$

`sc` The community abundance of the scaling community

`species_params` A data.frame to hold the species specific parameters (see the mizer vignette, Table 2, for details)

`interaction` The species specific interaction matrix, θ_{ij}

`srr` Function to calculate the realised (density dependent) recruitment. Has two arguments which are rdi and species_params

`selectivity` An array (gear x species x w) that holds the selectivity of each gear for species and size, $S_{g,i,w}$

`catchability` An array (gear x species) that holds the catchability of each species by each gear, $Q_{g,i}$

`initial_n` An array (species x size) that holds abundance of each species at each weight at our candidate steady state solution.

`initial_n_pp` A vector the same length as the w_full slot that describes the abundance of the background background resource at each weight.

`n` Exponent of maximum intake rate.

`p` Exponent of metabolic cost.

`lambda` Exponent of resource spectrum.

`q` Exponent for volumetric search rate.

`f0` Initial feeding level.

`kappa` Magnitude of resource spectrum.

`A` Abundance multipliers.

`linecolour` A named vector of colour values, named by species. Used to give consistent colours to species in plots.

`linetype` A named vector of linetypes, named by species. Used to give consistent colours to species in plots.

Note

The MizerParams class is fairly complex with a large number of slots, many of which are multidimensional arrays. The dimensions of these arrays is strictly enforced so that `MizerParams` objects are consistent in terms of number of species and number of size classes.

Although it is possible to build a `MizerParams` object by hand it is not recommended and several constructors are available.

The `MizerParams` class does not hold any dynamic information, e.g. abundances or harvest effort through time. These are held in MizerSim objects.

See Also

`project` `MizerSim`

`MizerSim` *Constructor for the* `MizerSim` *class*

Description

A constructor for the `MizerSim` class. This is used by the `project` method to create `MizerSim` objects of the right dimensions. It is not necessary for users to use this constructor.

Usage

```
MizerSim(object, ...)

## S4 method for signature 'MizerParams'
MizerSim(object, t_dimnames = NA, t_max = 100,
  t_save = 1)
```

Arguments

`object`	a MizerParams object
`...`	Other arguments (currently not used).
`t_dimnames`	Numeric vector that is used for the time dimensions of the slots. Default = NA.
`t_max`	The maximum time step of the simulation. Only used if t_dimnames = NA. Default value = 100.
`t_save`	How often should the results of the simulation be stored. Only used if t_dimnames = NA. Default value = 1.

Value

An object of type MizerSim

See Also

`project` MizerParams MizerSim

Examples

```
## Not run:
data(NS_species_params_gears)
data(inter)
params <- MizerParams(NS_species_params_gears, inter)
sim <- project(params)

## End(Not run)
```

`MizerSim-class` *MizerSim*

Description

A class that holds the results of projecting a MizerParams object through time.

Details

MizerSim objects are created by using the `project` method on an object of type `MizerParams`.

There are several plotting methods available to explore the contents of a `MizerSim` object. See the package vignette for more details.

Slots

`params` An object of type MizerParams.

`n` Array that stores the projected community population abundances by time, species and size

`effort` Array that stores the fishing effort through time by time and gear

`n_pp` Array that stores the projected background population by time and size

See Also

`project` `MizerParams`

NS_species_params *Example parameter set based on the North Sea*

Description

This data set is based on species in the North Sea (Blanchard et al.). It is a data.frame that contains all the necessary information to be used by the `MizerParams` constructor. As there is no gear column, each species is assumed to be fished by a separate gear.

Format

A data frame with 12 rows and 7 columns. Each row is a species.

species Name of the species

w_inf The von Bertalanffy W_infinity parameter

w_mat Size at maturity

beta Size preference ratio

sigma Width of the size-preference

r_max Maximum recruitment

k_vb The von Bertalanffy k parameter

Source

Blanchard et al.

NS_species_params_gears *Example parameter set based on the North Sea with different gears*

Description

This data set is based on species in the North Sea (Blanchard et al.). It is similar to the data set `NS_species_params` except that this one has an additional column specifying the fishing gear that operates on each species.

Format

A data frame with 12 rows and 8 columns. Each row is a species.

species Name of the species

w_inf The von Bertalanffy W_infinity parameter

w_mat Size at maturity

beta Size preference ratio

sigma Width of the size-preference

r_max Maximum recruitment

k_vb The von Bertalanffy k parameter

gear Name of the fishing gear

Source

Blanchard et al.

`plot,MizerSim,missing-method`
Summary plot for `MizerSim` *objects*

Description

After running a projection, produces 5 plots in the same window: feeding level, abundance spectra, predation mortality and fishing mortality of each species by size; and biomass of each species through time. This method just uses the other plotting methods and puts them all in one window.

Usage

```
## S4 method for signature 'MizerSim,missing'
plot(x, y, ...)
```

Arguments

`x`	An object of class MizerSim
`y`	Not used
`...`	For additional arguments see the documentation for `plotBiomass`, `plotFeedingLevel`,`plotSpectra`,`plo` and `plotFMort`.

Value

A viewport object

Examples

```
## Not run:
data(NS_species_params_gears)
data(inter)
params <- MizerParams(NS_species_params_gears, inter)
sim <- project(params, effort=1, t_max=20, t_save = 2)
plot(sim)
plot(sim, time_range = 10:20) # change time period for size-based plots
plot(sim, min_w = 10, max_w = 1000) # change size range for biomass plot

## End(Not run)
```

plotBiomass *Plot the biomass of species through time*

Description

After running a projection, the biomass of each species can be plotted against time. The biomass is calculated within user defined size limits (min_w, max_w, min_l, max_l, see getBiomass).

Usage

```
plotBiomass(sim, ...)

## S4 method for signature 'MizerSim'
plotBiomass(sim,
  species = sim@params@species_params$species[!is.na(sim@params@A)],
  start_time = as.numeric(dimnames(sim@n)[[1]][1]),
  end_time = as.numeric(dimnames(sim@n)[[1]][dim(sim@n)[1]]),
  y_ticks = 6, print_it = TRUE, ylim = c(NA, NA), total = FALSE,
  background = TRUE, ...)
```

Arguments

`sim`	An object of class MizerSim
`...`	Other arguments to pass to `getBiomass` method, for example `min_w` and `max_w`
`species`	Name or vector of names of the species to be plotted. By default all species are plotted.
`start_time`	The first time to be plotted. Default is the beginning of the time series.
`end_time`	The last time to be plotted. Default is the end of the time series.
`y_ticks`	The approximate number of ticks desired on the y axis
`print_it`	Display the plot, or just return the ggplot2 object. Default value is TRUE
`ylim`	A numeric vector of length two providing limits of for the y axis. Use NA to refer to the existing minimum or maximum. Any values below 1e-20 are always cut off.
`total`	A boolean value that determines whether the total biomass from all species is plotted as well. Default is FALSE
`background`	A boolean value that determines whether background species are included. Ignored if the model does not contain background species. Default is TRUE.

Value

A ggplot2 object

See Also

getBiomass

Examples

```
## Not run:
data(NS_species_params_gears)
data(inter)
params <- MizerParams(NS_species_params_gears, inter)
sim <- project(params, effort=1, t_max=20, t_save = 0.2)
plotBiomass(sim)
plotBiomass(sim, species = c("Cod", "Herring"), total = TRUE)
plotBiomass(sim, min_w = 10, max_w = 1000)
plotBiomass(sim, start_time = 10, end_time = 15)
plotBiomass(sim, y_ticks = 3)

## End(Not run)
```

`plotFeedingLevel` *Plot the feeding level of species by size*

Description

After running a projection, plot the feeding level of each species by size. The feeding level is averaged over the specified time range (a single value for the time range can be used).

Usage

```
plotFeedingLevel(sim, ...)

## S4 method for signature 'MizerSim'
plotFeedingLevel(sim,
  species = as.character(sim@params@species_params$species),
  time_range = max(as.numeric(dimnames(sim@n)$time)), print_it = TRUE,
  ...)
```

Arguments

`sim`	An object of class MizerSim.
`...`	Other arguments to pass to `getFeedingLevel` method
`species`	Name or vector of names of the species to be plotted. By default all species are plotted.
`time_range`	The time range (either a vector of values, a vector of min and max time, or a single value) to average the abundances over. Default is the final time step.
`print_it`	Display the plot, or just return the ggplot2 object. Defaults to TRUE

Value

A ggplot2 object

See Also

getFeedingLevel

Examples

```
## Not run:
data(NS_species_params_gears)
data(inter)
params <- MizerParams(NS_species_params_gears, inter)
sim <- project(params, effort=1, t_max=20, t_save = 2)
plotFeedingLevel(sim)
plotFeedingLevel(sim, time_range = 10:20)

## End(Not run)
```

---

plotFMort *Plot total fishing mortality of each species by size*

Description

After running a projection, plot the total fishing mortality of each species by size. The total fishing mortality is averaged over the specified time range (a single value for the time range can be used to plot a single time step).

Usage

```
plotFMort(sim, ...)

## S4 method for signature 'MizerSim'
plotFMort(sim,
  species = as.character(sim@params@species_params$species),
  time_range = max(as.numeric(dimnames(sim@n)$time)), print_it = TRUE,
  ...)
```

Arguments

sim	An object of class MizerSim.
...	Other arguments to pass to getFMort method
species	Name or vector of names of the species to be plotted. By default all species are plotted.
time_range	The time range (either a vector of values, a vector of min and max time, or a single value) to average the abundances over. Default is the final time step.
print_it	Display the plot, or just return the ggplot2 object. Defaults to TRUE

Value

A ggplot2 object

See Also

getFMort

Examples

```
## Not run:
data(NS_species_params_gears)
data(inter)
params <- MizerParams(NS_species_params_gears, inter)
sim <- project(params, effort=1, t_max=20, t_save = 2)
plotFMort(sim)
plotFMort(sim, time_range = 10:20)

## End(Not run)
```

---

plotGrowthCurves *Plot growth curves giving weight as a function of age*

Description

If given a MizerSim object, uses the growth rates at the final time of a simulation to calculate the size at age. If given a MizerParams object, uses the initial growth rates instead.

Usage

```
plotGrowthCurves(object, ...)

## S4 method for signature 'MizerSim'
plotGrowthCurves(object,
  species = as.character(sim@params@species_params$species),
  max_age = 20, percentage = FALSE, print_it = TRUE)

## S4 method for signature 'MizerParams'
plotGrowthCurves(object,
  species = as.character(params@species_params$species), max_age = 20,
  percentage = FALSE, print_it = TRUE)
```

Arguments

`object`	MizerSim or MizerParams object
`...`	Other arguments (unused)
`species`	Name or vector of names of the species to be plotted. By default all species are plotted.
`max_age`	The age up to which the weight is to be plotted. Default is 20
`percentage`	Boolean value. If TRUE, the size is shown as a percentage of the maximal size.
`print_it`	Display the plot, or just return the ggplot2 object. Defaults to TRUE

Details

When the growth curve for only a single species is plotted, horizontal lines are included that indicate the maturity size and the maximum size for that species. If furthermore the species parameters contain the variables a and b for length to weight conversion and the von Bertalanffy parameter k_vb, then the von Bertalanffy growth curve is superimposed in black.

Value

A ggplot2 object

Examples

```
## Not run:
data(NS_species_params_gears)
data(inter)
params <- MizerParams(NS_species_params_gears, inter)
sim <- project(params, effort=1, t_max=20, t_save = 2)
plotGrowthCurves(sim, percentage = TRUE)
plotGrowthCurves(sim, species = "Cod", max_age = 24)

## End(Not run)
```

---

plotM2 *Plot predation mortality rate of each species against size*

Description

After running a projection, plot the predation mortality rate of each species by size. The mortality rate is averaged over the specified time range (a single value for the time range can be used to plot a single time step).

Usage

```
plotM2(sim, ...)

## S4 method for signature 'MizerSim'
plotM2(sim,
  species = as.character(sim@params@species_params$species),
  time_range = max(as.numeric(dimnames(sim@n)$time)), print_it = TRUE,
  ...)
```

Arguments

sim	An object of class MizerSim
...	Other arguments to pass to getM2 method.
species	Name or vector of names of the species to be plotted. By default all species are plotted.

`time_range`	The time range (either a vector of values, a vector of min and max time, or a single value) to average the abundances over. Default is the final time step.
`print_it`	Display the plot, or just return the ggplot2 object. Defaults to TRUE

Value

A ggplot2 object

See Also

`getM2`

Examples

```
## Not run:
data(NS_species_params_gears)
data(inter)
params <- MizerParams(NS_species_params_gears, inter)
sim <- project(params, effort=1, t_max=20, t_save = 2)
plotM2(sim)
plotM2(sim, time_range = 10:20)

## End(Not run)
```

`plotSpectra` *Plot the abundance spectra*

Description

What is plotted is the number density multiplied by a power of the weight, with the power specified by the `power` argument.

Usage

```
plotSpectra(object, ...)

## S4 method for signature 'MizerSim'
plotSpectra(object, species = NULL,
  time_range = max(as.numeric(dimnames(object@n)$time)),
  min_w = min(object@params@w)/100, ylim = c(NA, NA), power = 1,
  biomass = TRUE, print_it = TRUE, total = FALSE, plankton = TRUE,
  background = TRUE, ...)

## S4 method for signature 'MizerParams'
plotSpectra(object, species = NULL,
  min_w = min(object@w)/100, ylim = c(NA, NA), power = 1,
  biomass = TRUE, print_it = TRUE, total = FALSE, plankton = TRUE,
  background = TRUE, ...)
```

Arguments

`object`	An object of class MizerSim or MizerParams.
`...`	Other arguments (currently unused)
`species`	Name or vector of names of the species to be plotted. By default all species are plotted.
`time_range`	The time range (either a vector of values, a vector of min and max time, or a single value) to average the abundances over. Default is the final time step. Ignored when called with a MizerParams object.
`min_w`	Minimum weight to be plotted (useful for truncating the background spectrum). Default value is a hundredth of the minimum size value of the community.
`ylim`	A numeric vector of length two providing limits of for the y axis. Use NA to refer to the existing minimum or maximum. Any values below 1e-20 are always cut off.
`power`	The abundance is plotted as the number density times the weight raised to `power`. The default `power = 1` gives the biomass density, whereas `power = 2` gives the biomass density with respect to logarithmic size bins.
`biomass`	Obsolete. Only used if `power` argument is missing. Then `biomass = TRUE` is equivalent to `power=1` and `biomass = FALSE` is equivalent to `power=0`
`print_it`	Display the plot, or just return the ggplot2 object. Defaults to TRUE
`total`	A boolean value that determines whether the total over all species in the system is plotted as well. Default is FALSE
`plankton`	A boolean value that determines whether plankton is included. Default is TRUE.
`background`	A boolean value that determines whether background species are included. Ignored if the model does not contain background species. Default is TRUE.

Details

When called with a MizerSim object, the abundance is averaged over the specified time range (a single value for the time range can be used to plot a single time step). When called with a MizerParams object the initial abundance is plotted.

Value

A ggplot2 object

Examples

```
## Not run:
data(NS_species_params_gears)
data(inter)
params <- MizerParams(NS_species_params_gears, inter)
sim <- project(params, effort=1, t_max=20, t_save = 2)
plotSpectra(sim)
plotSpectra(sim, min_w = 1e-6)
plotSpectra(sim, time_range = 10:20)
plotSpectra(sim, time_range = 10:20, power = 0)
plotSpectra(sim, species = c("Cod", "Herring"), power = 1)

## End(Not run)
```

---

plotYield *Plot the total yield of species through time*

Description

After running a projection, the total yield of each species across all fishing gears can be plotted against time. The yield is obtained with `getYield`.

Usage

```
plotYield(sim, sim2, ...)

## S4 method for signature 'MizerSim,missing'
plotYield(sim,
  species = as.character(sim@params@species_params$species),
  print_it = TRUE, total = FALSE, log = TRUE, ...)

## S4 method for signature 'MizerSim,MizerSim'
plotYield(sim, sim2,
  species = as.character(sim@params@species_params$species),
  print_it = TRUE, total = FALSE, log = TRUE, ...)
```

Arguments

`sim`	An object of class MizerSim
`sim2`	An optional second object of class MizerSim. If this is provided its yields will be shown on the same plot in bolder lines.
`...`	Other arguments to pass to `getYield` method
`species`	Name or vector of names of the species to be plotted. By default all species contained in `sim` are plotted.
`print_it`	Display the plot, or just return the ggplot2 object. Defaults to TRUE
`total`	A boolean value that determines whether the total yield from all species in the system is plotted as well. Default is FALSE
`log`	Boolean whether yield should be plotted on a logarithmic axis. Defaults to true.

Value

A ggplot2 object

See Also

`getYield`

Examples

```
## Not run:
data(NS_species_params_gears)
data(inter)
params <- MizerParams(NS_species_params_gears, inter)
sim <- project(params, effort=1, t_max=20, t_save = 0.2)
plotYield(sim)
plotYield(sim, species = c("Cod", "Herring"), total = TRUE)

# Comparing with yield from twice the effort
sim2 <- project(params, effort=2, t_max=20, t_save = 0.2)
plotYield(sim, sim2, species = c("Cod", "Herring"), log = FALSE)

## End(Not run)
```

`plotYieldGear` *Plot the total yield of each species by gear through time*

Description

After running a projection, the total yield of each species by fishing gear can be plotted against time.

Usage

```
plotYieldGear(sim, ...)

## S4 method for signature 'MizerSim'
plotYieldGear(sim,
  species = as.character(sim@params@species_params$species),
  print_it = TRUE, total = FALSE, ...)
```

Arguments

`sim`	An object of class MizerSim
`...`	Other arguments to pass to `getYieldGear` method
`species`	Name or vector of names of the species to be plotted. By default all species are plotted.
`print_it`	Display the plot, or just return the ggplot2 object. Defaults to TRUE
`total`	A boolean value that determines whether the total yield per gear over all species in the system is plotted as well. Default is FALSE

Details

This plot is pretty easy to do by hand. It just gets the biomass using the `getYieldGear` method and plots using the ggplot2 package. You can then fiddle about with colours and linetypes etc. Just look at the source code for details.

Value

A ggplot2 object

See Also

`getYieldGear`

Examples

```
## Not run:
data(NS_species_params_gears)
data(inter)
params <- MizerParams(NS_species_params_gears, inter)
sim <- project(params, effort=1, t_max=20, t_save = 0.2)
plotYieldGear(sim)
plotYieldGear(sim, species = c("Cod", "Herring"), total = TRUE)

## End(Not run)
```

`project` *project method for the size based modelling*

Description

Runs the size-based model simulation and projects the size based model through time. `project` is called using an object of type MizerParams and an object that contains the effort of the fishing gears through time. The method returns an object of type MizerSim which can then be explored with a range of summary and plotting methods.

Usage

```
project(object, effort, ...)

## S4 method for signature 'MizerParams,missing'
project(object, effort, ...)

## S4 method for signature 'MizerParams,numeric'
project(object, effort, t_max = 100,
  dt = 0.1, ...)

## S4 method for signature 'MizerParams,array'
project(object, effort, t_save = 1,
  dt = 0.1, initial_n = object@initial_n,
  initial_n_pp = object@initial_n_pp, shiny_progress = NULL, ...)
```

Arguments

`object`	A MizerParams object
`effort`	The effort of each fishing gear through time. See notes below.
`...`	Currently unused.
`t_max`	The maximum time the projection runs for. The default value is 100. However, this argument is not needed if an array is used for the `effort` argument, in which case this argument is ignored. See notes below.
`dt`	Time step of the solver. The default value is 0.1.
`t_save`	The frequency with which the output is stored. The default value is 1. Must be an integer multiple of dt.
`initial_n`	The initial populations of the species. By default the `initial_n` slot of the MizerParams argument is used. See the notes below.
`initial_n_pp`	The initial population of the background spectrum. It should be a numeric vector of the same length as the `w_full` slot of the `MizerParams` argument. By default the `initial_n_pp` slot of the MizerParams argument is used.
`shiny_progress`	A shiny progress object used to update shiny progress bar. Default NULL.

Value

An object of type `MizerSim`.

Note

The `effort` argument specifies the level of fishing effort during the simulation. It can be specified in three different ways:

- A single numeric value. This specifies the effort of all fishing gears which is constant through time (i.e. all the gears have the same constant effort).
- A numerical vector which has the same length as the number of fishing gears. The vector must be named and the names must correspond to the gear names in the `MizerParams` object. The values in the vector specify the constant fishing effort of each of the fishing gears, i.e. the effort is constant through time but each gear may have a different fishing effort.
- A numerical array with dimensions time step x gear. This specifies the fishing effort of each gear at each time step. The first dimension, time, must be named numerically and contiguously. The second dimension of the array must be named and the names must correspond to the gear names in the `MizerParams` argument. The value for the effort for a particular time is used during the interval from that time to the next time in the array.

If effort is specified as an array then the smallest time in the array is used as the initial time for the simulation. Otherwise the initial time is set to 0. Also, if the effort is an array then the `t_max` argument is ignored and the maximum simulation time is the largest time of the effort array.

The `initial_n` argument is a matrix with dimensions species x size. It specifies the abundances of the species at the initial time. The order of species must be the same as in the `MizerParams` argument. If the initial population is not specified, the argument is set by default by the `get_initial_n` function which is set up for a North Sea model.

See Also

MizerParams

Examples

```
## Not run:
# Data set with different fishing gears
data(NS_species_params_gears)
data(inter)
params <- MizerParams(NS_species_params_gears, inter)
# With constant fishing effort for all gears for 20 time steps
sim <- project(params, t_max = 20, effort = 0.5)
# With constant fishing effort which is different for each gear
effort <- c(Industrial = 0, Pelagic = 1, Beam = 0.5, Otter = 0.5)
sim <- project(params, t_max = 20, effort = effort)
# With fishing effort that varies through time for each gear
gear_names <- c("Industrial","Pelagic","Beam","Otter")
times <- seq(from = 1, to = 10, by = 1)
effort_array <- array(NA, dim = c(length(times), length(gear_names)),
    dimnames = list(time = times, gear = gear_names))
effort_array[,"Industrial"] <- 0.5
effort_array[,"Pelagic"] <- seq(from = 1, to = 2, length = length(times))
effort_array[,"Beam"] <- seq(from = 1, to = 0, length = length(times))
effort_array[,"Otter"] <- seq(from = 1, to = 0.5, length = length(times))
sim <- project(params, effort = effort_array)

## End(Not run)
```

project_methods *Methods used for projecting*

Description

The methods defined in the file project_methods calculate the various quantities needed to project the size-spectra forward in time, using the model described in section 3 of the mizer vignette.

List of Methods

In this list we relate the methods in this file to the quantities named in the mizer vignette.

Method name	Expression	Description	Section in vignet
getPhiPrey	$E_{a.i}(w)$	Available energy	3.2
getFeedingLevel	$f_i(w)$	Feeding level	3.3
getPredRate	$\phi_i(w_p/w)(1-f_i(w))\gamma_i w^q N_i(w) dw$	Predation	3.7
getM2	$\mu_{p.i}(w)$	Predation mortality	3.7
getM2Background	$\mu_p(w)$	Predation mortality on background	3.8
getFMortGear	$F_{g,i}(w)$	Fishing mortality by gear	8.3
getFMort	$\mu_{f.i}(w)$	Total fishing mortality	8.3
getZ	$\mu_i(w)$	Total mortality	3.7

getEReproAndGrowth	$E_{r.i}(w)$	Energy put into growth and reproduction	3.4
getESpawning	$\psi_i(w)E_{r.i}(w)$	Energy put reproduction	3.5
getEGrowth	$g_i(w)$	Energy put growth	3.4
getRDI	$R_{p.i}$	Egg production	3.5
getRDD	R_i	Recruitment	3.6

retune_abundance *Retunes abundance of background species.*

Description

An unexported helper function.

Usage

```
retune_abundance(params, retune)
```

Arguments

params	A MizerParams object
retune	A boolean vector that determines whether a species can be retuned or not.

Details

If N_i(w) is a steady state of the McKendrik-von Foerster (MVF) equation with fixed growth and death rates, then A_i*N_i(w) is also a steady state, where A_i is an abundance multiplier. When we add a foreground species to our model, we want to choose new abundance multipliers of the background species so that the community abundance after adding the new species is close to the original community abundance, stored in `params@sc`.

Value

An object of type `MizerParams`

See Also

MizerParams

setBackground *Designate species as background species*

Description

Background species are handled differently in some plots and their abundance is automatically adjusted in addSpecies() to keep the community close to the Sheldon spectrum.

Usage

```
setBackground(object, ...)

## S4 method for signature 'MizerParams'
setBackground(object,
  species = object@species_params$species)

## S4 method for signature 'MizerSim'
setBackground(object,
  species = object@params@species_params$species)
```

Arguments

`object`	An object of class MizerParams or MizerSim.
`...`	Other arguments (unused)
`species`	Name or vector of names of the species to be designated as background species. By default this is set to all species.

Value

An object of the same class as the `object` argument

Examples

```
## Not run:
data(NS_species_params_gears)
data(inter)
params <- MizerParams(NS_species_params_gears, inter)
sim <- project(params, effort=1, t_max=20, t_save = 0.2)
sim <- setBackground(sim, species = c("Sprat", "Sandeel",
                                           "N.pout", "Dab", "Saithe"))
plotSpectra(sim)

## End(Not run)
```

set_community_model *Sets up parameters for a community-type model*

Description

This functions creates a `MizerParams` object so that community-type models can be easily set up and run. A community model has several features that distinguish it from the food-web type models. Only one 'species' is resolved, i.e. one 'species' is used to represent the whole community. The resource spectrum only extends to the start of the community spectrum. Recruitment to the smallest size in the community spectrum is constant and set by the user. As recruitment is constant, the proportion of energy invested in reproduction (the slot `psi` of the returned `MizerParams` object) is set to 0. Standard metabolism has been turned off (the parameter `ks` is set to 0). Consequently, the growth rate is now determined solely by the assimilated food (see the package vignette for more details).

Usage

```
set_community_model(max_w = 1e+06, min_w = 0.001, z0 = 0.1,
  alpha = 0.2, h = 10, beta = 100, sigma = 2, q = 0.8, n = 2/3,
  kappa = 1000, lambda = 2 + q - n, f0 = 0.7, r_pp = 10,
  gamma = NA, knife_edge_size = 1000, knife_is_min = TRUE,
  recruitment = kappa * min_w^-lambda, rec_mult = 1, ...)
```

Arguments

`max_w`	The maximum size of the community. The `w_inf` of the species used to represent the community is set to 0.9 * this value. The default value is 1e6.
`min_w`	The minimum size of the community. Default value is 1e-3.
`z0`	The background mortality of the community. Default value is 0.1.
`alpha`	The assimilation efficiency of the community. Default value 0.2
`h`	The maximum food intake rate. Default value is 10.
`beta`	The preferred predator prey mass ratio. Default value is 100.
`sigma`	The width of the prey preference. Default value is 2.0.
`q`	The search volume exponent. Default value is 0.8.
`n`	The scaling of the intake. Default value is 2/3.
`kappa`	The carrying capacity of the background spectrum. Default value is 1000.
`lambda`	The exponent of the background spectrum. Default value is 2 + q - n.
`f0`	The average feeding level of individuals who feed mainly on the resource. This value is used to calculate the search rate parameter `gamma` (see the package vignette). Default value is 0.7.
`r_pp`	Growth rate of the primary productivity. Default value is 10.
`gamma`	Volumetric search rate. Estimated using `h`, `f0` and `kappa` if not supplied.

`knife_edge_size`	The size at the edge of the knife-selectivity function. Default value is 1000.
`knife_is_min`	Is the knife-edge selectivity function selecting above (TRUE) or below (FALSE) the edge. Default is TRUE.
`recruitment`	The constant recruitment in the smallest size class of the community spectrum. This should be set so that the community spectrum continues the background spectrum. Default value = `kappa` * `min_w^-lambda`.
`rec_mult`	Additional multiplier for the constant recruitment. Default value is 1.
`...`	Other arguments to pass to the `MizerParams` constructor.

Details

The function has many arguments, all of which have default values. The main arguments that the users should be concerned with are `z0`, `recruitment`, `alpha` and `f0` as these determine the average growth rate of the community.

Fishing selectivity is modelled as a knife-edge function with one parameter, `knife_edge_size`, which determines the size at which species are selected.

The resulting `MizerParams` object can be projected forward using `project()` like any other `MizerParams` object. When projecting the community model it may be necessary to reduce `dt` to 0.1 to avoid any instabilities with the solver. You can check this by plotting the biomass or abundance through time after the projection.

Value

An object of type `MizerParams`

References

K. H. Andersen,J. E. Beyer and P. Lundberg, 2009, Trophic and individual efficiencies of size-structured communities, Proceedings of the Royal Society, 276, 109-114

Examples

```
## Not run:
params <- set_community_model(f0=0.7, z0=0.2, recruitment=3e7)
sim <- project(params, effort = 0, t_max = 100, dt=0.1)
plotBiomass(sim)
plotSpectra(sim)

## End(Not run)
```

set_scaling_model *Sets up parameters for a scale free trait-based model*

Description

This functions creates a `MizerParams` object so that scale free trait-based-type models can be easily set up and run. The scale free trait-based size spectrum model can be derived as a simplification of the general size-based model used in `mizer`. All the species-specific parameters are the same for all species, except for the egg size, maturity size and asymptotic size. These differ over the species, but the ratio of egg size to maturity size and the ratio of egg size to asymptotic size are the same for each species. The asymptotic sizes of the species are spread evenly on a logarithmic scale. See the `mizer` vignette and the Details section below for more details and examples of the scale free trait-based model.

Usage

```
set_scaling_model(no_sp = 11, min_w_inf = 10, max_w_inf = 10^3,
  min_egg = 10^(-4), min_w_mat = 10^(0.4),
  no_w = log10(max_w_inf/min_egg) * 100 + 1, min_w_pp = min_egg/(beta *
  exp(5 * sigma)), n = 2/3, q = 3/4, lambda = 2 + q - n,
  r_pp = 0.1, kappa = 0.005, alpha = 0.4, ks = 4, h = 30,
  beta = 100, sigma = 1.3, f0 = 0.6, knife_edge_size = 100,
  gear_names = "knife_edge_gear", rfac = Inf, ...)
```

Arguments

`no_sp`	The number of species in the model. The default value is 11.
`min_w_inf`	The asymptotic size of the smallest species in the community. Default value is 10.
`max_w_inf`	The asymptotic size of the largest species in the community. Default value is 1000.
`min_egg`	The size of the the egg of the smallest species. Default value is 10^(-4).
`min_w_mat`	The maturity size of the smallest species. Default value is 10^(0.4),
`no_w`	The number of size bins in the community spectrum. Default value is such that there are 100 bins for each factor of 10 in weight.
`min_w_pp`	The smallest size of the background spectrum. Default value is min_egg/(beta*exp(5*sigma)) so that it covers the entire range of the feeding kernel of even the smallest fish larva.
`n`	Scaling of the intake. Default value is 2/3.
`q`	Exponent of the search volume. Default value is 3/4 unless `lambda` is provided, in which case this argument is ignored and q = lambda - 2 + n.
`lambda`	Exponent of the abundance power law. If supplied, this overrules the `q` argument. Otherwise the default value is 2+q-n.
`r_pp`	Growth rate of the primary productivity. Default value is 0.1.

`kappa`	Coefficient in abundance power law. Default value is 0.005.
`alpha`	The assimilation efficiency of the community. The default value is 0.4.
`ks`	Standard metabolism coefficient. Default value is 4.
`h`	Maximum food intake rate. Default value is 30.
`beta`	Preferred predator prey mass ratio. Default value is 100.
`sigma`	Width of prey size preference. Default value is 1.3.
`f0`	Expected average feeding level. Used to set `gamma`, the coefficient in the search rate. The default value is 0.6.
`knife_edge_size`	The minimum size at which the gear or gears select species. Must be of length 1 or no_sp. Default value is 100.
`gear_names`	The names of the fishing gears. A character vector, the same length as the knife_edge_size parameter. Default value is "knife_edge_gear".
`rfac`	The factor such that Rmax = rfac * R, where Rmax is the maximum recruitment allowed and R is the steady-state recruitment. Thus the larger `rfac` the less the impact of the non-linear stock-recruitment curve. The default is Inf.
`...`	Other arguments to pass to the `MizerParams` constructor.

Details

The scale free trait-based model is similar to the standard trait-based model, with three main differences:

1. We have an exact equation for a steady state of this system which is often stable, even when we include no extra stabilization effects like density dependence or stock recruitment relationships.
2. The egg size is proportional to the maturity size for each species
3. The parameters are chosen so that R_0 (the expected number of offspring produced by an individual over a lifetime) is close to 1 for each species.

The function has many arguments, all of which have default values. Of particular interest to the user are the number of species in the model and the minimum and maximum asymptotic sizes.

The characteristic weights of the different species are defined by min_egg, min_w_mat, min_w_inf, max_w_inf and no_sp, in the sense that the egg weights of the no_sp species are logarithmically evenly spaced, ranging from min_w=min_egg to max_w=max_w_inf. The maturity weights of the species can be obtained by multiplying the egg_weights by min_w_mat/min_egg. The asymptotic weights of the species can be obtained by multiplying the egg weights by min_w_inf/min_egg.

Although the scale free trait based model's default steady state is often stable without imposing a stock recruitment relationship, the function can set a Beverton-Holt type stock recruitment relationship that imposes a maximal reproduction rate that is a multiple of the recruitment rate at steady state. That multiple is set by the argument `rfac`.

In addition to setting up the parameters, this function also evaluates the analytic expression for a steady state of the scale free trait-based model and sets it as the initial condition.

The search rate coefficient `gamma` is calculated using the expected feeding level, `f0`.

The option of including fishing is given, but the steady state may lose its natural stability if too much fishing is included. In such a case the user may wish to include stabilizing effects (like Rmax and chi) to ensure the steady state is stable. Fishing selectivity is modelled as a knife-edge function with one parameter, `knife_edge_size`, which is the size at which species are selected. Each species can either be fished by the same gear (`knife_edge_size` has a length of 1) or by a different gear (the length of `knife_edge_size` has the same length as the number of species and the order of selectivity size is that of the asymptotic size).

The resulting `MizerParams` object can be projected forward using `project()` like any other `MizerParams` object. When projecting the model it may be necessary to reduce `dt` to 0.1 to avoid any instabilities with the solver. You can check this by plotting the biomass or abundance through time after the projection.

Value

An object of type `MizerParams`

See Also

MizerParams

Examples

```
## Not run:
s_params <- set_scaling_model()
sim <- project(s_params, t_max=5, effort = 0)
plotSpectra(sim)

## End(Not run)
```

`set_trait_model` *Sets up parameters for a trait-based model*

Description

This functions creates a `MizerParams` object so that trait-based-type models can be easily set up and run. The trait-based size spectrum model can be derived as a simplification of the general size-based model used in `mizer`. All the species-specific parameters are the same, except for the asymptotic size, which is considered the most important trait characterizing a species. Other parameters are related to the asymptotic size. For example, the size at maturity is given by w_inf * eta, where eta is the same for all species. For the trait-based model the number of species is not important. For applications of the trait-based model see Andersen & Pedersen (2010). See the `mizer` vignette for more details and examples of the trait-based model.

Usage

```
set_trait_model(no_sp = 10, min_w_inf = 10, max_w_inf = 1e+05,
  no_w = 100, min_w = 0.001, max_w = max_w_inf * 1.1,
  min_w_pp = 1e-10, no_w_pp = NA, w_pp_cutoff = 1, k0 = 50,
  n = 2/3, p = 0.75, q = 0.9, eta = 0.25, r_pp = 4,
  kappa = 0.005, lambda = 2 + q - n, alpha = 0.6, ks = 4,
  z0pre = 0.6, h = 30, beta = 100, sigma = 1.3, f0 = 0.5,
  gamma = NA, knife_edge_size = 1000, gear_names = "knife_edge_gear",
  ...)
```

Arguments

`no_sp`	The number of species in the model. The default value is 10. The more species, the longer takes to run.
`min_w_inf`	The asymptotic size of the smallest species in the community.
`max_w_inf`	The asymptotic size of the largest species in the community.
`no_w`	The number of size bins in the community spectrum.
`min_w`	The smallest size of the community spectrum.
`max_w`	The largest size of the community spectrum. Default value is the largest w_inf in the community x 1.1.
`min_w_pp`	The smallest size of the background spectrum.
`no_w_pp`	Obsolete argument that is no longer used because the number of plankton size bins is determined because all size bins have to be logarithmically equally spaced.
`w_pp_cutoff`	The cut off size of the background spectrum. Default value is 1.
`k0`	Multiplier for the maximum recruitment. Default value is 50.
`n`	Scaling of the intake. Default value is 2/3.
`p`	Scaling of the standard metabolism. Default value is 0.75.
`q`	Exponent of the search volume. Default value is 0.9.
`eta`	Factor to calculate `w_mat` from asymptotic size.
`r_pp`	Growth rate of the primary productivity. Default value is 4.
`kappa`	Coefficient in abundance power law. Default value is 0.005.
`lambda`	Exponent of the abundance power law. Default value is (2+q-n).
`alpha`	The assimilation efficiency of the community. The default value is 0.6
`ks`	Standard metabolism coefficient. Default value is 4.
`z0pre`	The coefficient of the background mortality of the community. z0 = z0pre * w_inf ^ (n-1). The default value is 0.6.
`h`	Maximum food intake rate. Default value is 30.
`beta`	Preferred predator prey mass ratio. Default value is 100.
`sigma`	Width of prey size preference. Default value is 1.3.
`f0`	Expected average feeding level. Used to set `gamma`, the factor for the search volume. The default value is 0.5.

`gamma`	Volumetric search rate. Estimated using `h`, `f0` and `kappa` if not supplied.
`knife_edge_size`	The minimum size at which the gear or gears select species. Must be of length 1 or no_sp.
`gear_names`	The names of the fishing gears. A character vector, the same length as the number of species. Default is 1 - no_sp.
`...`	Other arguments to pass to the `MizerParams` constructor.

Details

The function has many arguments, all of which have default values. Of particular interest to the user are the number of species in the model and the minimum and maximum asymptotic sizes. The asymptotic sizes of the species are spread evenly on a logarithmic scale within this range.

The stock recruitment relationship is the default Beverton-Holt style. The maximum recruitment is calculated using equilibrium theory (see Andersen & Pedersen, 2010) and a multiplier, `k0`. Users should adjust `k0` to get the spectra they want.

The factor for the search volume, `gamma`, is calculated using the expected feeding level, `f0`.

Fishing selectivity is modelled as a knife-edge function with one parameter, `knife_edge_size`, which is the size at which species are selected. Each species can either be fished by the same gear (`knife_edge_size` has a length of 1) or by a different gear (the length of `knife_edge_size` has the same length as the number of species and the order of selectivity size is that of the asymptotic size).

The resulting `MizerParams` object can be projected forward using `project()` like any other `MizerParams` object. When projecting the community model it may be necessary to reduce `dt` to 0.1 to avoid any instabilities with the solver. You can check this by plotting the biomass or abundance through time after the projection.

Value

An object of type `MizerParams`

References

K. H. Andersen and M. Pedersen, 2010, Damped trophic cascades driven by fishing in model marine ecosystems. Proceedings of the Royal Society V, Biological Sciences, 1682, 795-802.

See Also

MizerParams

Examples

```
## Not run:
trait_params <- set_trait_model(no_sp = 15)
init_pop <- get_initial_n(trait_params, n0_mult = 0.001)
sim <- project(trait_params, effort = 0, t_max = 50, dt=0.2,
    initial_n = init_pop, t_save = 1)
plot(sim)
```

```
## Set up industrial fishery that only fishes on species with w_inf <= 500 g
## And where the selectivity of the industrial fishery = w_inf * 0.05
no_sp <- 10
min_w_inf <- 10
max_w_inf <- 1e5
w_inf <- 10^seq(from=log10(min_w_inf), to = log10(max_w_inf), length=no_sp)
knife_edges <- w_inf * 0.05
industrial_gears <- w_inf <= 500
other_gears <- w_inf > 500
gear_names <- rep("Industrial", no_sp)
gear_names[other_gears] <- "Other"
params_gear <- set_trait_model(no_sp = no_sp, min_w_inf = min_w_inf,
    max_w_inf = max_w_inf, knife_edge_size = knife_edges,
    gear_names = gear_names)
## Only turn on Industrial fishery. Set effort of the Other gear to 0
sim <- project(params_gear, t_max = 20, effort = c(Industrial = 1, Other = 0))

## End(Not run)
```

sigmoid_length *Length based sigmoid selectivity function*

Description

A sigmoid shaped selectivity function. Based on two parameters l25 and l50 which determine the length at which 25% and 50% of the stock is selected respectively. As the size-based model is weight based, and this selectivity function is length based, it is also necessary to supply the length-weight parameters a and b.

Usage

```
sigmoid_length(w, l25, l50, a, b)
```

Arguments

w	the size of the individual.
l25	the length which gives a selectivity of 25%.
l50	the length which gives a selectivity of 50%.
a	the multiplier of the length-weight function.
b	the exponent of the length-weight function.

steady *Tune params object to be at steady state*

Description

Tune params object to be at steady state

Usage

```
steady(params, rfac = Inf, effort = 0)
```

Arguments

`params`	A MizerParams object
`rfac`	A number that determines the strength of the non-linearity in the Beverton-Holt stock-recruitment relationship. The maximal recruitment will be set to rfac times the normal steady-state recruitment. Default value is 10.
`effort`	The fishing effort. Default is 0

summary,MizerParams-method *Summarize MizerParams object*

Description

Outputs a general summary of the structure and content of the object

Usage

```
## S4 method for signature 'MizerParams'
summary(object, ...)
```

Arguments

`object`	A `MizerParams` object.
`...`	Other arguments (currently not used).

Examples

```
## Not run:
data(NS_species_params_gears)
data(inter)
params <- MizerParams(NS_species_params_gears,inter)
summary(params)

## End(Not run)
```

`summary,MizerSim-method` *Summarize MizerSim object*

Description

Outputs a general summary of the structure and content of the object

Usage

```
## S4 method for signature 'MizerSim'
summary(object, ...)
```

Arguments

`object`	A `MizerSim` object.
`...`	Other arguments (currently not used).

Examples

```
## Not run:
data(NS_species_params_gears)
data(inter)
params <- MizerParams(NS_species_params_gears,inter)
sim <- project(params, effort=1, t_max=5)
summary(sim)

## End(Not run)
```

`wrapper_functions` *Functions used for setting up models*

Description

The functions defined in the file wrapper functions set up MizerParams objects for various kinds of size-spectrum models.

List of Functions

In this list we relate the functions in this file to the sections in the mizer vignette where the corresponding model is described.

Function name	Description	Section in vignette
`set_community_model`	Community model	5
`set_trait_model`	Trait-based model	6
`set_scaling_model`	Scale-invariant Trait-based model	7

The file also contains a helper function `retune_abundance`.

Index

addSpecies, 3
addSpecies,MizerParams-method (addSpecies), 3

display_frames, 4, *6*, *28*

get_initial_n, 32
getBiomass, 5, *6*, *42*
getBiomass,MizerSim-method (getBiomass), 5
getBiomassFrame, 6
getBiomassFrame,MizerSim-method (getBiomassFrame), 6
getCommunitySlope, 7
getCommunitySlope,MizerSim-method (getCommunitySlope), 7
getEGrowth, 8, *54*
getEGrowth,MizerParams,matrix,numeric,matrix,matrix-method (getEGrowth), 8
getEGrowth,MizerParams,matrix,numeric,missing,missing-method (getEGrowth), 8
getEReproAndGrowth, *8*, 9, *11*, *54*
getEReproAndGrowth,MizerParams,matrix,numeric,matrix-method (getEReproAndGrowth), 9
getEReproAndGrowth,MizerParams,matrix,numeric,missing-method (getEReproAndGrowth), 9
getESpawning, *8*, 10, *27*, *54*
getESpawning,MizerParams,matrix,numeric,matrix-method (getESpawning), 10
getESpawning,MizerParams,matrix,numeric,missing-method (getESpawning), 10
getFeedingLevel, *10*, 11, *23*, *44*, *53*
getFeedingLevel,MizerParams,matrix,numeric,matrix-method (getFeedingLevel), 11
getFeedingLevel,MizerParams,matrix,numeric,missing-method (getFeedingLevel), 11
getFeedingLevel,MizerSim,missing,missing,missing-method (getFeedingLevel), 11
getFMort, 13, *31*, *45*, *53*
getFMort,MizerParams,matrix-method (getFMort), 13
getFMort,MizerParams,numeric-method (getFMort), 13
getFMort,MizerSim,missing-method (getFMort), 13
getFMortGear, 14, *53*
getFMortGear,MizerParams,matrix-method (getFMortGear), 14
getFMortGear,MizerParams,numeric-method (getFMortGear), 14
getFMortGear,MizerSim,missing-method (getFMortGear), 14
getM2, 16, *18*, *23*, *31*, *47*, *53*
getM2,MizerParams,matrix,numeric,missing-method (getM2), 16
getM2,MizerParams,missing,missing,array-method (getM2), 16
getM2,MizerSim,missing,missing,missing-method (getM2), 16
getM2Background, 18, *53*
getM2Background,MizerParams,matrix,numeric,array-method (getM2Background), 18
getM2Background,MizerParams,matrix,numeric,missing-method (getM2Background), 18
getMeanMaxWeight, 19
getMeanMaxWeight,MizerSim-method (getMeanMaxWeight), 19
getMeanWeight, 20
getMeanWeight,MizerSim-method (getMeanWeight), 20
getN, 21
getN,MizerSim-method (getN), 21
getPhiPrey, *12*, 22, *53*
getPhiPrey,MizerParams,matrix,numeric-method (getPhiPrey), 22
getPredRate, *17*, 23, *53*
getPredRate,MizerParams,matrix,numeric,matrix-method (getPredRate), 23

getPredRate,MizerParams,matrix,numeric,missing-method
  (getPredRate), 23
getProportionOfLargeFish, 24
getProportionOfLargeFish,MizerSim-method
  (getProportionOfLargeFish), 24
getRDD, 25, *54*
getRDD,MizerParams,matrix,numeric,matrix-method
  (getRDD), 25
getRDD,MizerParams,matrix,numeric,missing-method
  (getRDD), 25
getRDI, *26*, 26, *54*
getRDI,MizerParams,matrix,numeric,matrix-method
  (getRDI), 26
getRDI,MizerParams,matrix,numeric,missing-method
  (getRDI), 26
getSSB, 27, *29*
getSSB,MizerSim-method (getSSB), 27
getSSBFrame, 28
getSSBFrame,MizerSim-method
  (getSSBFrame), 28
getYield, 29, *30*, *49*
getYield,MizerSim-method (getYield), 29
getYieldGear, *29*, 30, *50*, *51*
getYieldGear,MizerSim-method
  (getYieldGear), 30
getZ, 30, *53*
getZ,MizerParams,matrix,numeric,numeric,matrix-method
  (getZ), 30
getZ,MizerParams,matrix,numeric,numeric,missing-method
  (getZ), 30

inter, 32

knife_edge, 33

log_breaks, 33

mizer, 34
mizer, (mizer), 34
mizer-package (mizer), 34
MizerParams, *3*, *8*, *22*, *23*, *32*, *34*, 34, *36*,
  38–40, *45*, *48*, *51–57*, *60*, *62*, *64*
MizerParams,data.frame,matrix-method
  (MizerParams), 34
MizerParams,data.frame,missing-method
  (MizerParams), 34
MizerParams,numeric,missing-method
  (MizerParams), 34
MizerParams-class, 36
MizerSim, *6*, *28*, *36*, *38*, 38, *39*, *41–46*, *48–51*,
  55
MizerSim,MizerParams-method (MizerSim),
  38
MizerSim-class, 39

NS_species_params, 40
NS_species_params_gears, 40

plot,MizerSim,missing-method, 41
plotBiomass, *41*, 42
plotBiomass,MizerSim-method
  (plotBiomass), 42
plotFeedingLevel, *41*, 43
plotFeedingLevel,MizerSim-method
  (plotFeedingLevel), 43
plotFMort, *41*, 44
plotFMort,MizerSim-method (plotFMort),
  44
plotGrowthCurves, 45
plotGrowthCurves,MizerParams-method
  (plotGrowthCurves), 45
plotGrowthCurves,MizerSim-method
  (plotGrowthCurves), 45
plotM2, *41*, 46
plotM2,MizerSim-method (plotM2), 46
plotSpectra, *41*, 47
plotSpectra,MizerParams-method
  (plotSpectra), 47
plotSpectra,MizerSim-method
  (plotSpectra), 47
plotYield, 49
plotYield,MizerSim,missing-method
  (plotYield), 49
plotYield,MizerSim,MizerSim-method
  (plotYield), 49
plotYieldGear, 50
plotYieldGear,MizerSim-method
  (plotYieldGear), 50
project, *8–11*, *16–18*, *22*, *23*, *27*, *36*, *38*, *39*,
  51
project,MizerParams,array-method
  (project), 51
project,MizerParams,missing-method
  (project), 51
project,MizerParams,numeric-method
  (project), 51
project_methods, 53

retune_abundance, 54, *66*

set_community_model, 56, *65*
set_scaling_model, 58, *65*
set_trait_model, 60, *65*
setBackground, 55
setBackground,MizerParams-method
        *(*setBackground*)*, 55
setBackground,MizerSim-method
        *(*setBackground*)*, 55
sigmoid_length, 63
steady, 64
summary,MizerParams-method, 64
summary,MizerSim-method, 65

wrapper_functions, 65