

# Package ‘mizer’

May 2, 2018

**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

**License** GPL-3

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

**LinkingTo** Rcpp

**Depends** R (>= 3.4)

**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** 6.0.1

**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** 2018-05-02 13:21:07

## 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 . . . . .	15
getM2 . . . . .	16
getM2Background . . . . .	18
getMeanMaxWeight . . . . .	19
getMeanWeight . . . . .	20
getN . . . . .	21
getPhiPrey . . . . .	22
getPredRate . . . . .	23
getProportionOfLargeFish . . . . .	24
getRDD . . . . .	25
getRDI . . . . .	27
getSSB . . . . .	28
getSSBFrame . . . . .	29
getYield . . . . .	30
getYieldGear . . . . .	31
getZ . . . . .	32
get_initial_n . . . . .	33
inter . . . . .	34
knife_edge . . . . .	34
log_breaks . . . . .	35
mizer . . . . .	35
MizerParams . . . . .	36
MizerParams-class . . . . .	38
MizerSim . . . . .	40
MizerSim-class . . . . .	41
NS_species_params . . . . .	41
NS_species_params_gears . . . . .	42
plot,MizerSim,missing-method . . . . .	43
plotBiomass . . . . .	44
plotFeedingLevel . . . . .	45
plotFMort . . . . .	46
plotGrowthCurves . . . . .	47
plotM2 . . . . .	48
plotSpectra . . . . .	49
plotYield . . . . .	51
plotYieldGear . . . . .	52
project . . . . .	54
project_methods . . . . .	56
retune_abundance . . . . .	56

*addSpecies* 3

setBackground . . . . .	57
set_community_model . . . . .	58
set_scaling_model . . . . .	60
set_trait_model . . . . .	63
sigmoid_length . . . . .	65
steady . . . . .	66
summary,MizerParams-method . . . . .	66
summary,MizerSim-method . . . . .	67
wrapper_functions . . . . .	68

**Index** 69

---

*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.  
Add species to a MizerParams object

### Usage

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

### Arguments

<code>params</code>	A mizer params object for the original system.
<code>...</code>	Other arguments (unused)
<code>species_params</code>	The species parameters of the new species we want to add to the system.
<code>SSB</code>	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.
<code>rfac</code>	A number that determines the strength of the non-linearity in the Beverton-Holt stock-recruitment relationship. The maximal recruitment will be set to <code>rfac</code> times the normal steady-state recruitment. Default value is 10.
<code>effort</code>	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	<i>Calculate the total biomass of each species within a size range at each time step.</i>
------------	---

---

**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).

Calculate the total biomass of each species from a MizerSim object

**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](#)).

Get the biomass frame from a MizerSim object.

### 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 <a href="#">MizerSim</a>
...	Other arguments to pass to <a href="#">getBiomass</a> 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.

Calculate the slope of the community abundance from a `MizerSim` object.

### Usage

```
getCommunitySlope(object, ...)

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

### Arguments

<code>object</code>	An object of class <code>MizerSim</code> .
<code>...</code>	Optional parameters include <ul style="list-style-type: none"> <li>• <code>min_w</code> Minimum weight of species to be used in the calculation.</li> <li>• <code>max_w</code> Maximum weight of species to be used in the calculation.</li> <li>• <code>min_l</code> Minimum length of species to be used in the calculation.</li> <li>• <code>max_l</code> Maximum length of species to be used in the calculation.</li> </ul>
<code>species</code>	Numeric or character vector of species to include in the calculation.
<code>biomass</code>	Boolean. If <code>TRUE</code> (default), the abundance is based on biomass, if <code>FALSE</code> 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.

getEGrowth method with e\_spawning and e arguments.

getEGrowth method without e\_spawning and e arguments.

### 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 <a href="#">MizerParams</a> 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 <a href="#">getESpawning</a> 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 <a href="#">getEReproAndGrowth</a> 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.

getEReproAndGrowth method with feeding\_level argument.

getEReproAndGrowth method without feeding\_level argument.

**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.

getESpawning method with e argument.

getESpawning method without e argument.

**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	<i>Get feeding level</i>
-----------------	--------------------------

---

**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.

getFeedingLevel method for a MizerParams object with already calculated phi\_preymatrix.

getFeedingLevel method for a MizerParams object without the phi\_preymatrix argument.

getFeedingLevel method for a MizerSim object.

**Usage**

```

getFeedingLevel(object, n, n_pp, phi_pre, ...)

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

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

## 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_pre	The PhiPrey matrix (optional) of dimension no. species x no. size bins. If not passed in, it is calculated internally using the <a href="#">getPhiPrey</a> 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	<i>Get the total fishing mortality rate from all fishing gears by time, species and size.</i>
----------	---

---

**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,  $\mu_{f,i}(w) = \sum_g F_{g,i,w}$ .

getFMort method for MizerParams object with constant effort.

getFMort method for MizerParams object with time changing effort.

getFMort method for MizerSim object.

**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.

getFMortGear method for MizerParams object with constant effort.

getFMortGear method for MizerParams object with time changing effort.

getFMortGear method for MizerSim object.

### 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)
```



**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.

getM2 method for MizerParams object with pred\_rate argument.

getM2 method for MizerParams object without pred\_rate argument.

getM2 method for MizerSim object.

**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.

getM2Background method with pred\_array argument.

getM2Background method without pred\_array argument.

**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.

Calculate the mean maximum weight of the community from a MizerSim object.

**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.

Calculate the mean weight of the community from a MizerSim object.

**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	<i>Calculate the total abundance in terms of numbers of species within a size range</i>
------	---

---

**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)

Calculate the total abundance from a MizerSim object

**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](#)).

getPredRate method with feeding\_level argument.

getPredRate method without feeding\_level argument.

**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 <code>getFeedingLevel()</code> 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.

Calculate the proportion of large fish from a `MizerSim` object.



**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)

```

**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.

getRDD method with rdi argument.

getRDD method without rdi argument.

**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 <a href="#">getRDI</a> 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	<i>Get density independent recruitment</i>
--------	--

---

### 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.

getRDI method with `e_spawning` argument.

getRDI method without `e_spawning` argument.

### 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

<code>object</code>	A <code>MizerParams</code> object.
<code>n</code>	A matrix of species abundance (species x size).
<code>n_pp</code>	A vector of the background abundance by size.
<code>e_spawning</code>	The energy available for spawning (optional). A matrix of size no. species x no. size bins. If not supplied, is calculated internally using the <a href="#">getESpawning</a> method.
<code>...</code>	Other arguments (currently unused).
<code>sex_ratio</code>	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.

Calculate the SSB from a MizerSim object

**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	<i>Get data frame of spawning stock biomass of species through time, ready for ggplot2</i>
-------------	--

---

### Description

After running a projection, the spawning stock biomass of each species can be plotted against time. Get the biomass frame from a `MizerSim` object.

### 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

<code>sim</code>	An object of class <code>MizerSim</code>
<code>...</code>	Other arguments to pass to <code>getSBB</code> method, currently unused.
<code>species</code>	Name or vector of names of the species to be plotted. By default all species are plotted.
<code>start_time</code>	The first time to be plotted. Default is the beginning of the time series.
<code>end_time</code>	The last time to be plotted. Default is the end of the time series.
<code>ylim</code>	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.
<code>total</code>	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	<i>Calculate the total yield of each species</i>
----------	--

---

### Description

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

Calculate the total yield of each species from a MizerSim object

### 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	<i>Calculate the total yield per gear and species</i>
--------------	---

---

**Description**

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

Calculate the total yield per gear from a MizerSim object

**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.

getZ method with m2 argument.

getZ method without m2 argument.

### 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	<i>Calculate initial population abundances for the community populations</i>
---------------	--

---

**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 <a href="#">MizerParams</a> .
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.

**Usage**

inter

**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.

**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

Constructor that takes the species\_params data.frame and the interaction matrix

Constructor based on the species\_params data.frame only with no interaction

**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 calculate 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 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,  $\theta_{ij}$   
**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.

MizerSim constructor taking only a MizerParams object.

**Usage**

```
MizerSim(object, ...)
```

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

**Arguments**

<code>object</code>	a <a href="#">MizerParams</a> object
<code>...</code>	Other arguments (currently not used).
<code>t_dimnames</code>	Numeric vector that is used for the time dimensions of the slots. Default = NA.
<code>t_max</code>	The maximum time step of the simulation. Only used if <code>t_dimnames = NA</code> . Default value = 100.
<code>t_save</code>	How often should the results of the simulation be stored. Only used if <code>t_dimnames = NA</code> . 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	<i>MizerSim</i>
----------------	-----------------

---

## 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	<i>Example parameter set based on the North Sea</i>
-------------------	---

---

## 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.

## Usage

NS\_species\_params

**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.

**Usage**

NS\_species\_params\_gears

**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 <a href="#">MizerSim</a>
y	Not used
...	For additional arguments see the documentation for <a href="#">plotBiomass</a> , <a href="#">plotFeedingLevel</a> , <a href="#">plotSpectra</a> , <a href="#">plotPredationMortality</a> and <a href="#">plotFMort</a> .

## 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](#)).

Plot the biomass using a `MizerSim` object.

### 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

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

### 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	<i>Plot the feeding level of species by size</i>
------------------	--

---

**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).

Plot the feeding level using a MizerSim object.

**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 <a href="#">MizerSim</a> .
...	Other arguments to pass to <code>getFeedingLevel</code> 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).

Plot total fishing mortality using a MizerSim object.

**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 <a href="#">MizerSim</a> .
...	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.

Plot growth curves using a MizerSim object.

Plot growth curves using a MizerSim object.

**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).

Plot M2 using a MizerSim object.



**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 <a href="#">MizerSim</a>
...	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.

Plot the abundance spectra using a MizerSim object.

Plot the abundance spectra using a MizerParams object.

**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 <a href="#">MizerSim</a> or <a href="#">MizerParams</a> .
...	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 <a href="#">MizerParams</a> 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](#).

Plot the yield using a MizerSim object.

Plot the yield using two MizerSim objects.

**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 <a href="#">MizerSim</a>
sim2	An optional second object of class <a href="#">MizerSim</a> . If this is provided its yields will be shown on the same plot in bolder lines.
...	Other arguments to pass to <code>getYield</code> method
species	Name or vector of names of the species to be plotted. By default all species contained in <code>sim</code> are plotted.
print_it	Display the plot, or just return the <code>ggplot2</code> 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. Plot the yield of each gear using a `MizerSim` object.

**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 <a href="#">MizerSim</a>
...	Other arguments to pass to <code>getYieldGear</code> 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 <code>ggplot2</code> 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.

Project without an effort argument.

Project with a constant effort.

Project with time varying effort

## 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

<code>object</code>	A <code>MizerParams</code> object
<code>effort</code>	The effort of each fishing gear through time. See notes below.
<code>...</code>	Currently unused.
<code>t_max</code>	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 <code>effort</code> argument, in which case this argument is ignored. See notes below.
<code>dt</code>	Time step of the solver. The default value is 0.1.
<code>t_save</code>	The frequency with which the output is stored. The default value is 1. Must be an integer multiple of <code>dt</code> .
<code>initial_n</code>	The initial populations of the species. By default the <code>initial_n</code> slot of the <code>MizerParams</code> argument is used. See the notes below.
<code>initial_n_pp</code>	The initial population of the background spectrum. It should be a numeric vector of the same length as the <code>w_full</code> slot of the <code>MizerParams</code> argument. By default the <code>initial_n_pp</code> slot of the <code>MizerParams</code> 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 vignette
<a href="#">getPhiPrey</a>	$E_{a,i}(w)$	Available energy	3.2
<a href="#">getFeedingLevel</a>	$f_i(w)$	Feeding level	3.3
<a href="#">getPredRate</a>	$\phi_i(w_p/w)(1 - f_i(w))\gamma_i w^q N_i(w)dw$	Predation	3.7
<a href="#">getM2</a>	$\mu_{p,i}(w)$	Predation mortality	3.7
<a href="#">getM2Background</a>	$\mu_p(w)$	Predation mortality on background	3.8
<a href="#">getFMortGear</a>	$F_{g,i}(w)$	Fishing mortality by gear	8.3
<a href="#">getFMort</a>	$\mu_{f,i}(w)$	Total fishing mortality	8.3
<a href="#">getZ</a>	$\mu_i(w)$	Total mortality	3.7
<a href="#">getEReproAndGrowth</a>	$E_{r,i}(w)$	Energy put into growth and reproduction	3.4
<a href="#">getESpawning</a>	$\psi_i(w)E_{r,i}(w)$	Energy put reproduction	3.5
<a href="#">getEGrowth</a>	$g_i(w)$	Energy put growth	3.4
<a href="#">getRDI</a>	$R_{p,i}$	Egg production	3.5
<a href="#">getRDD</a>	$R_i$	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 McKendrick-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	<i>Designate species as background species</i>
---------------	--

---

**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.

Set species to background species in `MizerParams` object

Set species to background species in `MizerSim` object

**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 <a href="#">MizerParams</a> or <a href="#">MizerSim</a> .
...	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 function 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 $0.9 * \text{this value}$ . 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 = $\text{kappa} * \text{min\_w}^{-\text{lambda}}$ .
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	<i>Sets up parameters for a scale free trait-based model</i>
-------------------	--

---

## Description

This function 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 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 $\text{min\_egg}/(\text{beta}*\exp(5*\text{sigma}))$ 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 $\text{lambda}$ is provided, in which case this argument is ignored and $q = \text{lambda} - 2 + n$ .
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 $\text{gamma}$ , 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 <code>knife_edge_size</code> parameter. Default value is "knife_edge_gear".
rfac	The factor such that $R_{\text{max}} = \text{rfac} * R$ , where $R_{\text{max}}$ is the maximum recruitment allowed and $R$ is the steady-state recruitment. Thus the larger $\text{rfac}$ 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	<i>Sets up parameters for a trait-based model</i>
-----------------	---

---

### Description

This function 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  $w_{\text{inf}} * \text{eta}$ , 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 $w_{\text{inf}}$ 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 $w_{\text{mat}}$ 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. $z_0 = z_{0pre} * w_{inf}^{(n-1)}$ . 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,  $k_0$ . Users should adjust  $k_0$  to get the spectra they want.

The factor for the search volume, gamma, is calculated using the expected feeding level,  $f_0$ .

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 125 and 150 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, 125, 150, 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	<i>Tune params object to be at steady state</i>
--------	---

---

**Description**

Tune params object to be at steady state

**Usage**

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

**Arguments**

params	A <a href="#">MizerParams</a> 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	<i>Summarize MizerParams object</i>
----------------------------	-------------------------------------

---

**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
<a href="#">set_community_model</a>	Community model	5
<a href="#">set_trait_model</a>	Trait-based model	6
<a href="#">set_scaling_model</a>	Scale-invariant Trait-based model	7

The file also contains a helper function [retune\\_abundance](#).

# Index

## \*Topic **datasets**

- inter, [34](#)
- NS\_species\_params, [41](#)
- NS\_species\_params\_gears, [42](#)
  
- addSpecies, [3](#)
- addSpecies,MizerParams-method  
(addSpecies), [3](#)
  
- display\_frames, [4](#), [6](#), [29](#)
  
- get\_initial\_n, [33](#)
- getBiomass, [5](#), [6](#), [44](#), [45](#)
- getBiomass,MizerSim-method  
(getBiomass), [5](#)
- getBiomassFrame, [6](#)
- getBiomassFrame,MizerSim-method  
(getBiomassFrame), [6](#)
- getCommunitySlope, [7](#)
- getCommunitySlope,MizerSim-method  
(getCommunitySlope), [7](#)
- getEGrowth, [8](#), [56](#)
- getEGrowth,MizerParams,matrix,numeric,matrix,matrix-method  
(getEGrowth), [8](#)
- getEGrowth,MizerParams,matrix,numeric,missing,missing-method  
(getEGrowth), [8](#)
- getEReproAndGrowth, [8](#), [9](#), [11](#), [56](#)
- getEReproAndGrowth,MizerParams,matrix,numeric,matrix-method  
(getEReproAndGrowth), [9](#)
- getEReproAndGrowth,MizerParams,matrix,numeric,missing-method  
(getEReproAndGrowth), [9](#)
- getESpawning, [8](#), [10](#), [27](#), [56](#)
- getESpawning,MizerParams,matrix,numeric,matrix-method  
(getESpawning), [10](#)
- getESpawning,MizerParams,matrix,numeric,missing-method  
(getESpawning), [10](#)
- getFeedingLevel, [10](#), [11](#), [24](#), [46](#), [56](#)
- 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](#), [32](#), [47](#), [56](#)
- getFMort,MizerParams,matrix-method  
(getFMort), [13](#)
- getFMort,MizerParams,numeric-method  
(getFMort), [13](#)
- getFMort,MizerSim,missing-method  
(getFMort), [13](#)
- getFMortGear, [15](#), [56](#)
- getFMortGear,MizerParams,matrix-method  
(getFMortGear), [15](#)
- getFMortGear,MizerParams,numeric-method  
(getFMortGear), [15](#)
- getFMortGear,MizerSim,missing-method  
(getFMortGear), [15](#)
- getM2, [16](#), [19](#), [23](#), [24](#), [32](#), [49](#), [56](#)
- 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](#), [56](#)
- 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](#), [56](#)

- getPhiPrey, MizerParams, matrix, numeric-method (getPhiPrey), 22
- getPredRate, 18, 23, 56
- 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, 56
- getRDD, MizerParams, matrix, numeric, matrix-method (getRDD), 25
- getRDD, MizerParams, matrix, numeric, missing-method (getRDD), 25
- getRDI, 26, 27, 56
- getRDI, MizerParams, matrix, numeric, matrix-method (getRDI), 27
- getRDI, MizerParams, matrix, numeric, missing-method (getRDI), 27
- getSSB, 28, 29
- getSSB, MizerSim-method (getSSB), 28
- getSSBFrame, 29
- getSSBFrame, MizerSim-method (getSSBFrame), 29
- getYield, 30, 31, 51, 52
- getYield, MizerSim-method (getYield), 30
- getYieldGear, 30, 31, 53
- getYieldGear, MizerSim-method (getYieldGear), 31
- getZ, 32, 56
- getZ, MizerParams, matrix, numeric, numeric, matrix-method (getZ), 32
- getZ, MizerParams, matrix, numeric, numeric, missing-method (getZ), 32
- inter, 34
- knife\_edge, 34
- log\_breaks, 35
- mizer, 35
- mizer, (mizer), 35
- mizer-package (mizer), 35
- MizerParams, 3, 8, 22–24, 33, 36, 36, 38–41, 47, 50, 51, 54, 55, 57–59, 62, 65, 66
- MizerParams, data.frame, matrix-method (MizerParams), 36
- MizerParams, data.frame, missing-method (MizerParams), 36
- MizerParams, numeric, missing-method (MizerParams), 36
- MizerParams-class, 38
- MizerSim, 6, 29, 37, 39, 40, 40, 41, 43–47, 49–54, 58
- MizerSim, MizerParams-method (MizerSim), 40
- MizerSim-class, 41
- NS\_species\_params, 41
- NS\_species\_params\_gears, 42
- plot, MizerSim, missing-method, 43
- plotBiomass, 43, 44
- plotBiomass, MizerSim-method (plotBiomass), 44
- plotFeedingLevel, 43, 45
- plotFeedingLevel, MizerSim-method (plotFeedingLevel), 45
- plotFMort, 43, 46
- plotFMort, MizerSim-method (plotFMort), 46
- plotGrowthCurves, 47
- plotGrowthCurves, MizerParams-method (plotGrowthCurves), 47
- plotGrowthCurves, MizerSim-method (plotGrowthCurves), 47
- plotM2, 43, 48
- plotM2, MizerSim-method (plotM2), 48
- plotSpectra, 43, 49
- plotSpectra, MizerParams-method (plotSpectra), 49
- plotSpectra, MizerSim-method (plotSpectra), 49
- plotYield, 51
- plotYield, MizerSim, missing-method (plotYield), 51
- plotYield, MizerSim, MizerSim-method (plotYield), 51
- plotYieldGear, 52
- plotYieldGear, MizerSim-method (plotYieldGear), 52
- project, 8–11, 17–19, 22–24, 27, 37–41, 54
- project, MizerParams, array-method (project), 54
- project, MizerParams, missing-method (project), 54

project,MizerParams,numeric-method  
    (project), [54](#)  
project\_methods, [56](#)

retune\_abundance, [56](#), [68](#)

set\_community\_model, [58](#), [68](#)  
set\_scaling\_model, [60](#), [68](#)  
set\_trait\_model, [63](#), [68](#)  
setBackground, [57](#)  
setBackground,MizerParams-method  
    (setBackground), [57](#)  
setBackground,MizerSim-method  
    (setBackground), [57](#)  
sigmoid\_length, [65](#)  
steady, [66](#)  
summary,MizerParams-method, [66](#)  
summary,MizerSim-method, [67](#)

wrapper\_functions, [68](#)