

Package: Rpath (via r-universe)

October 28, 2026

Type Package

Title R implementation of Ecopath with Ecosim

Version 1.2.0

Description This package implements the core mass balance equations made popular by Ecopath with Ecosim. There are two parts to the package, the static snapshot of energy flow through the system using rpath and dynamic simulations run using the rsim family of functions.

License file LICENSE

Encoding UTF-8

Imports data.table, methods, graphics, grDevices, MASS, Rcpp (>= 0.11.3), stats, utils, ggrepel, ggplot2, janitor, xml2

LinkingTo Rcpp

Suggests here, knitr, rmarkdown, distillery, ggpubr, rlist, stringr, viridis, qpdf, usethis, dplyr, generics, DT, testthat (>= 3.0.0)

VignetteBuilder knitr

URL <https://noaa-edab.github.io/Rpath/>, <https://github.com/NOAA-EDAB/Rpath/>

BugReports <https://github.com/NOAA-EDAB/Rpath/issues>

Depends R (>= 4.0.0)

Config/testthat/edition 3

LazyData true

Config/roxygen2/version 8.1.1

Config/pak/sysreqs libicu-dev libxml2-dev

Repository <https://noaa-fisheries-integrated-toolbox.r-universe.dev>

Date/Publication 2026-10-28 17:21:29 UTC

RemoteUrl <https://github.com/NOAA-EDAB/Rpath>

RemoteRef HEAD

RemoteSha 97871f34e6a19f4634c1abc51cd011ffb17c9f58

Contents

AB.params	3
adjust.fishing	4
adjust.forcing	5
adjust.scenario	6
check.rpath.params	7
create.rpath.from.eiixml	8
create.rpath.params	9
Ecosense.EBS	9
Ecosense.ECS	10
Ecosense.GOA	11
extract.node	12
fitcatch.to.forcecatch	13
frate.table	14
get.pp.link	15
get.rsim.fishing	16
get.rsim.forcing	16
get.rsim.params	17
get.rsim.stanzas	18
get.rsim.start_state	19
ggwebplot	19
import.eiixml	21
MTI	22
read.fitting.biomass	23
read.fitting.catch	23
read.rpath.params	24
REco.params	25
rpath	26
rpath.consumers	26
rpath.detrital	27
rpath.gears	28
rpath.groups	29
rpath.living	30
rpath.mixotrophs	30
rpath.producers	31
rpath.stanzas	32
rsim.deriv	33
rsim.diet	34
rsim.fishing	35
rsim.fit.apply	36
rsim.fit.list.bio.bysurvey	37
rsim.fit.list.bio.series	37
rsim.fit.list.catch.series	38
rsim.fit.obj	39
rsim.fit.obj.species	39
rsim.fit.remove.bio.timeseries	40
rsim.fit.run	41

rsim.fit.set.bio.wt	42
rsim.fit.set.catch.wt	43
rsim.fit.set.q	43
rsim.fit.table	44
rsim.fit.update	45
rsim.forcing	46
rsim.mort	47
rsim.params	47
rsim.plot	50
rsim.plot.fit	51
rsim.plot.fitspecies	52
rsim.predprey.table	53
rsim.run	54
rsim.scenario	55
rsim.sense	56
rsim.stanzas	57
rsim.state	59
rsim.step	60
set.rsim.scene	61
stanzaplot	62
webplot	63
write.Rpath	64
write.rpath.params	66
write.Rsim	66
Index	68

AB.params	<i>AB.params</i>
-----------	------------------

Description

Anchovy Bay (AB.params) is a sample ecosystem frequently used as an example ecosystem with EwE (see Christensen and Walters, 2024).

Usage

AB.params

Format

An unbalanced Rpath model object that contains:

- model** A data.table containing Ecopath unbalanced ecosystem parameters (base parameters and fisheries).
- diet** A data.table containing the Ecopath model’s diet matrix.
- stanzas** Parameters for multistanza groups.
- pedigree** A data.table containing the data quality (pedigree) for the Ecopath model. ...

Source

Christensen V, Walters CJ (2024) Ecosystem Modelling with EwE. The University of British Columbia, Vancouver, BC. doi:10.14288/24d7-ab68

Examples

```
# Balance the model
AB_bal <- rpath(AB.params)
```

adjust.fishing	<i>Adjust Fishing Mortality</i>
----------------	---------------------------------

Description

Modifies the fishing mortality value for a species by a particular gear. Parameters that can be adjusted using this function are: *ForcedEffort*, *ForcedFRate*, or *ForcedCatch*.

Usage

```
adjust.fishing(
  Rsim.scenario,
  parameter,
  group = NA,
  sim.year = 1,
  sim.month = 0,
  value
)
```

Arguments

Rsim.scenario	Scenario object that contains all of the rsim rates and forcing functions generated by <code>rsim.scenario()</code> .
parameter	Parameters to be modified (ForcedEffort,ForcedCatch,ForcedFRate)
group	Name of the group whose parameter is being changed. Valid values are found in the 'Group' field of the object created from running <code>rpath()</code>
sim.year	Year of the simulation that should be modified. Can be a range of years.
sim.month	Month of the year that should be modified. If set to 0, all months of the year are modified.
value	New value for the parameter.

Value

Returns an `Rsim.scenario()` object with the new fishing parameter values.

See Also

Other Adjust functions: [adjust.forcing\(\)](#), [adjust.scenario\(\)](#), [set.rsim.scene\(\)](#)

Examples

```
# Read in Rpath parameter file and generate balanced model
Rpath <- rpath(AB.params)
# Create a 50 yr Rsim scenario
Rsim.scenario <- rsim.scenario(Rpath, AB.params, years = 1:50)
# Change value of forcedFRate for Squids in years 3 through 5 to the value of 2 (for all months)
Rsim.scenario.adjusted.fishing <- adjust.fishing(Rsim.scenario, parameter = "ForcedFRate", group = "cod", sim.yea
head(Rsim.scenario.adjusted.fishing$fishing$ForcedFRate)
```

adjust.forcing	<i>Adjust Forcing Parameters</i>
----------------	----------------------------------

Description

Modifies the various forcing parameters of the rsim scenario object.

Usage

```
adjust.forcing(
  Rsim.scenario,
  parameter,
  group,
  sim.year = 1,
  sim.month = 0,
  bymonth = F,
  value
)
```

Arguments

Rsim.scenario	Scenario object that contains all of the rsim rates and forcing functions generated by rsim.scenario() .
parameter	Parameters to be modified (ForcedEffort, ForcedCatch, ForcedFRate)
group	Name of the group whose parameter is being changed. Valid values are found in the 'Group' field of the object created from running rpath()
sim.year	Year of the simulation that should be modified. Can be a range of years.
sim.month	Month of the year that should be modified. If set to 0, all months of the year are modified.
bymonth	Boolean value that denotes whether to use sim.year/sim.month combo or just sim.month as a sequential vector starting at 1.
value	New value for the parameter.

Value

Returns an Rsim.scenario object with the new parameter.

See Also

Other Adjust functions: [adjust.fishing\(\)](#), [adjust.scenario\(\)](#), [set.rsim.scene\(\)](#)

Examples

```
# Read in Rpath parameter file and generate balanced model
Rpath <- rpath(AB.params)
# Create a 50 yr Rsim scenario
Rsim.scenario <- rsim.scenario(Rpath, AB.params, years = 1:50)
# Adjust the ForcedPrey parameter for cod in year 1 for all months. Change the value to 10
Rsim.scenario.adjusted <- adjust.forcing(Rsim.scenario, parameter="ForcedPrey",group = "cod", sim.year = 1, sim.m
head(Rsim.scenario.adjusted$forcing$ForcedPrey)
```

adjust.scenario	<i>Adjust Rsim.scenario parameters</i>
-----------------	--

Description

Modifies the various parameters of the rsim.scenario() object. Parameters that can be adjusted using this function are: *B_BaseRef, MzeroMort, UnassimRespFrac, ActiveRespFrac, FtimeAdj, FtimeQB0pt, PB0pt, NoIntegrate, HandleSelf, ScrambleSelf, QQ, DD, VV, HandleSwitch, PredPredWeight, PreyPreyWeight*

Usage

```
adjust.scenario(Rsim.scenario, parameter, group, groupto = NA, value)
```

Arguments

Rsim.scenario	Scenario object that contains all of the rsim rates and forcing functions generated by rsim.scenario() .
parameter	Parameters to be modified (Choose from: <i>B_BaseRef, MzeroMort, UnassimRespFrac, ActiveRespFrac, FtimeAdj, FtimeQB0pt, PB0pt, NoIntegrate, HandleSelf, ScrambleSelf, QQ, DD, VV, HandleSwitch, PredPredWeight, PreyPreyWeight</i>)
group	The model group that the parameter change will affect. Note that a value of 'all' will affect all groups associated with the 'groupto' variable. Valid values are found in the 'Group' field of the object created from running rpath()
groupto	The corresponding group who's parameter is affecting the group variable. Required for parameters <i>QQ, DD, VV, HandleSwitch, PredPredWeight, PreyPreyWeight</i>
value	New value for the parameter.

Value

Returns an `rsim.scenario()` object with the new parameter.

See Also

Other Adjust functions: [adjust.fishing\(\)](#), [adjust.forcing\(\)](#), [set.rsim.scene\(\)](#)

Examples

```
# Read in Rpath parameter file and generate balanced model
Rpath <- rpath(AB.params)
# Create a 50 yr Rsim scenario
Rsim.scenario <- rsim.scenario(Rpath, AB.params, years = 1:50)
# Adjust the PBopt parameter for cod. Set to value = 2
Rsim.scenario.adjusted <- adjust.scenario(Rsim.scenario, parameter="PBopt", group = "cod", groupsto = "all", value = 2)
```

check.rpath.params	<i>Check Rpath parameter files</i>
--------------------	------------------------------------

Description

Logical check that the parameter files are filled out correctly, i.e. data is entered where it is expected.

Usage

```
check.rpath.params(Rpath.params)
```

Arguments

Rpath.params	R object containing the parameters needed to create a Rpath model. This is generated either by the create.rpath.params() or read.rpath.params() functions.
--------------	--

Value

Checks Rpath parameter files for consistency. An error message will be produced if one of the logical checks fails. Checks include: verification that all types are represented (e.g. consumer, producer, detrital, and fleet); check if input parameters were entered correctly; check if Diet columns sum to 1. (NOTE: This does not ensure data is correct just that it is in the right places).

See Also

Other Rpath functions: [MTI\(\)](#), [create.rpath.params\(\)](#), [frate.table\(\)](#), [read.rpath.params\(\)](#), [rpath\(\)](#), [rpath.consumers\(\)](#), [rpath.detrital\(\)](#), [rpath.gears\(\)](#), [rpath.groups\(\)](#), [rpath.living\(\)](#), [rpath.mixotrophs\(\)](#), [rpath.producers\(\)](#), [rpath.stanzas\(\)](#), [stanzaplot\(\)](#), [webplot\(\)](#), [write.Rpath\(\)](#), [write.rpath.params\(\)](#)

```
create.rpath.from.eiixml
```

Creates an Rpath object from an EwE exported model (EIIXML format)

Description

Parses an eiixml file exported using the Ecopath with Ecosim (EwE) GUI into an unbalanced Rpath model object, which can then be balanced. This function was tested on files exported by EwE version 6.7.

Usage

```
create.rpath.from.eiixml(eiifile, verbose = FALSE)
```

Arguments

eiifile	Path to exported EwE XML file
verbose	Logical. Use for debugging. If TRUE, prints out a list of parsed tables and number of rows read from the XML file

Value

An Rpath object (list) with the following components:

stanzas	Stanza parameters
pedigree	Pedigree parameters
diet	Diet parameters
model	Model parameters

See Also

Other xml: [import.eiixml\(\)](#)

Examples

```
# Import an eiixml file previously exported from the EwE GUI, check that the
# unbalanced model is functional, recalculate age-structured groups based on
# imported stanza parameters (if the imported model has stanzas), and balance
# the model.
eiixml <- system.file("extdata/xml/Western_Bering_Sea.eiixml", package = "Rpath")
rpath_object <- create.rpath.from.eiixml(eiixml)
check.rpath.params(rpath_object)
rpath_object <- rpath.stanzas(rpath_object)
rpath_balanced <- rpath(rpath_object)
```

create.rpath.params	<i>Creates a shell for the Rpath.params object</i>
---------------------	--

Description

Creates a shell of the Rpath.params list object which contains the *model*, *diet*, *multistanza*, and *pedigree* parameters.

Usage

```
create.rpath.params(group, type, stgroup = NA)
```

Arguments

group	Vector of group names.
type	Numeric vector of group type. Living = 0, Producer = 1, Detritus = 2, Fleet = 3.
stgroup	Vector of multistanza group names. Include NA for non-stanza groups.

Value

Outputs a list object of Rpath.params which are populated with values of NA or logical default values. Values can then be filled in using R. Use check.rpath.params() to ensure parameter files are filled out correctly (NOTE: This does not ensure data is correct just that it is in the right places).

See Also

Other Rpath functions: [MTI\(\)](#), [check.rpath.params\(\)](#), [frate.table\(\)](#), [read.rpath.params\(\)](#), [rpath\(\)](#), [rpath.consumers\(\)](#), [rpath.detrital\(\)](#), [rpath.gears\(\)](#), [rpath.groups\(\)](#), [rpath.living\(\)](#), [rpath.mixotrophs\(\)](#), [rpath.producers\(\)](#), [rpath.stanzas\(\)](#), [stanzaplot\(\)](#), [webplot\(\)](#), [write.Rpath\(\)](#), [write.rpath.params\(\)](#)

Ecosense.EBS	<i>Eastern Bering Sea 1990s Ecopath model</i>
--------------	---

Description

Rpath inputs for the eastern Bering Sea 1990s Ecopath model (53 biological groups and 1 fleet).

Usage

```
Ecosense.EBS
```

Format

An unbalanced Rpath model object that contains:

model A data.table containing Ecopath unbalanced ecosystem parameters (base parameters and fisheries).

diet A data.table containing the Ecopath model's diet matrix.

stanzas Parameters for multistanza groups.

pedigree A data.table containing the data quality (pedigree) for the Ecopath model. ...

Note

This is an aggregated version of the 1990s eastern Bering Sea Ecopath model of Aydin et al. (2007) and does not include stanzas.

Source

Whitehouse and Aydin 2020. Assessing the sensitivity of three Alaska marine food webs to perturbations: an example of Ecosim simulations using Rpath. <https://doi.org/10.1016/j.ecolmodel.2020.109074>

References

Aydin KY, Gaichas S, Ortiz I, Kinzey D, Friday N (2007) A comparison of the Bering Sea, Gulf of Alaska, and Aleutian Islands large marine ecosystems through food web modeling. U.S. Dep. Commer., NOAA Tech. Memo. NMFS-AFSC-178.

Examples

```
# Balance the model
ebs_bal <- rpath(Ecosense.EBS)
```

Ecosense.ECS

Eastern Chukchi Sea Ecopath model

Description

Rpath inputs for the eastern Chukchi Sea Ecopath model (52 biological groups and 1 fleet).

Usage

```
Ecosense.ECS
```

Format

An unbalanced Rpath model object that contains:

model A data.table containing Ecopath unbalanced ecosystem parameters (base parameters and fisheries).

diet A data.table containing the Ecopath model's diet matrix.

stanzas Parameters for multistanza groups.

pedigree A data.table containing the data quality (pedigree) for the Ecopath model. ...

Note

This is an aggregated version of the eastern Chukchi Sea Ecopath model of Whitehouse and Aydin (2016).

Source

Whitehouse and Aydin 2020. Assessing the sensitivity of three Alaska marine food webs to perturbations: an example of Ecosim simulations using Rpath. <https://doi.org/10.1016/j.ecolmodel.2020.109074>

References

Whitehouse GA, Aydin KY (2016) Trophic structure of the eastern Chukchi Sea: An updated mass balance food web model. U.S. Dep Commer, NOAA Tech. Memo. NMFS-AFSC-318. doi:10.7289/V5/TM-AFSC-318

Examples

```
# Balance the model
ecs_bal <- rpath(Ecosense.ECS)
```

Ecosense.GOA

Gulf of Alaska (west/central) 1990s Ecopath model

Description

Rpath inputs for the Gulf of Alaska (west/central) 1990s Ecopath model (49 biological groups and 1 fleet).

Usage

```
Ecosense.GOA
```

Format

An unbalanced Rpath model object that contains:

model A data.table containing Ecopath unbalanced ecosystem parameters (base parameters and fisheries).

diet A data.table containing the Ecopath model's diet matrix.

stanzas Parameters for multistanza groups.

pedigree A data.table containing the data quality (pedigree) for the Ecopath model. ...

Note

This is an aggregated version of the 1990s Gulf of Alaska Ecopath model of Aydin et al. (2007) and does not include stanzas.

Source

Whitehouse and Aydin 2020. Assessing the sensitivity of three Alaska marine food webs to perturbations: an example of Ecosim simulations using Rpath. <https://doi.org/10.1016/j.ecolmodel.2020.109074>

References

Aydin KY, Gaichas S, Ortiz I, Kinzey D, Friday N (2007) A comparison of the Bering Sea, Gulf of Alaska, and Aleutian Islands large marine ecosystems through food web modeling. U.S. Dep. Commer., NOAA Tech. Memo. NMFS-AFSC-178.

Examples

```
# Balance the model
goa_bal <- rpath(Ecosense.GOA)
```

extract.node	<i>Extract node/group data from an Rsim run</i>
--------------	---

Description

Extracts group/node specific information from a scenario run using `rsim.run()`

Usage

```
extract.node(Rsim.output, group)
```

Arguments

Rsim.output	Object generated by <code>rsim.run()</code> .
group	Name of the node/group for which you want to extract data. Valid values are found in the 'Group' field of the object created from running <code>rpath()</code>

Value

Returns a list object with monthly node specific data

Biomass	Vector of monthly Biomass values
AnnualBiomass	Vector of annual Biomass
TotalCatch	Vector of monthly Catch values
AnnualTotalCatch	Vector of annual Catch values
Landings	Dataframe (n x l) of monthly Landings. (Each column is a fishery that Lands the 'group')
Discards	Dataframe (n x d) of monthly Discards (Each column is a fishery that Discards the 'group')

See Also

Other Rsim functions: [rsim.params\(\)](#), [rsim.plot\(\)](#), [rsim.run\(\)](#), [rsim.scenario\(\)](#), [rsim.step\(\)](#), [write.Rsim\(\)](#)

Examples

```
# Read in Rpath parameter file and balance model
Rpath <- rpath(Ecosense.EBS)
# Create a 50 yr Rpath scenario
Rsim.scenario <- rsim.scenario(Rpath, Ecosense.EBS, years = 1:50)
# Run a scenario
Rsim.output <- rsim.run(Rsim.scenario, method = "RK4", years = 1:50)
# Extracts group (node) specific data from the simulation run
group <- extract.node(Rsim.output, Rpath$Group['Squids'])
```

fitcatch.to.forcecatch

Convert Fitting Catch to Forced Catch in Rsim Scenario

Description

Performs 3 steps: Sets effort to 0, sets default F, sets catch. Integrates observed catch data into an 'Rpath' simulation scene by zeroing out effort, setting a baseline fishing mortality rate, and forcing specific catches for observation years.

Usage

```
fitcatch.to.forcecatch(Rsim.scenario, Rpath)
```

Arguments

- Rsim.scenario An 'Rsim' scenario object containing the observed fishing data.
- Rpath A balanced 'Rpath' model object containing base biomass and landings.

Details

The function modifies the fishing parameters of the simulation by: 1. Setting all gear effort ('ForcedEffort') to zero. 2. Calculating a baseline Fishing Mortality rate (F) for all living and detrital groups based on the balanced 'Rpath' model ('(Landings + Discards) / Biomass'). 3. Assigning this calculated F to the 'ForcedFRate' matrix for all years. 4. Overriding specific years and groups with empirical catch data ('fitting\$Catch\$obs'). For these targeted overrides, the 'ForcedFRate' is set to 0 to allow the forced catch to drive the simulation.

Value

Modifies and returns the 'Rsim.scenario' object with updated 'ForcedEffort', 'ForcedFRate', and 'ForcedCatch' matrices.

frate.table	<i>Fishing Mortality Table</i>
-------------	--------------------------------

Description

Creates a table of fishing mortalities by species group and gear for an rsim.scenario() object.

Usage

```
frate.table(Rsim.scenario)
```

Arguments

- Rsim.scenario Scenario object that contains all of the rsim rates and forcing functions generated by `rsim.scenario()`.

Value

Returns a data table of F values for each species/gear combination.

Group	Group/node
Fishery/gearType	Gear type for the fishery
...	
Total	Sum over all gear types

See Also

Other Rpath functions: [MTI\(\)](#), [check.rpath.params\(\)](#), [create.rpath.params\(\)](#), [read.rpath.params\(\)](#), [rpath\(\)](#), [rpath.consumers\(\)](#), [rpath.detrital\(\)](#), [rpath.gears\(\)](#), [rpath.groups\(\)](#), [rpath.living\(\)](#), [rpath.mixotrophs\(\)](#), [rpath.producers\(\)](#), [rpath.stanzas\(\)](#), [stanzaplot\(\)](#), [webplot\(\)](#), [write.Rpath\(\)](#), [write.rpath.params\(\)](#)

Examples

```
# Read in Rpath parameter file and generate balanced model
Rpath <- rpath(AB.params)
# Create a 50 yr Rsim scenario
Rsim.scenario <- rsim.scenario(Rpath, AB.params, years = 1:50)
rates <- frate.table(Rsim.scenario)
# display the head of the data frame
head(rates)
```

get.pp.link

Get Predator-Prey Link Index

Description

Retrieves the numeric index of a specific predator-prey interaction from the model's internal diet network structure.

Usage

```
get.pp.link(Rsim.scenario, predator, prey)
```

Arguments

Rsim.scenario	An 'Rsim' scenario object containing the fitting data..
predator	A character string specifying the name of the predator group.
prey	A character string specifying the name of the prey group.

Details

This helper function scans the internal C++ 0-indexed arrays ('PreyFrom' and 'PreyTo'), converts them to 1-based R indices to look up the corresponding species names in 'sname', and returns the link index (or indices) where both the predator and prey match the requested strings.

Value

A numeric scalar (or vector) representing the internal link index associated with the specified predator-prey interaction.

get.rsim.fishing	<i>Retrieve fishing forcing parameters from an Rsim scenario</i>
------------------	--

Description

Helper function that will retrieve the fishing forcing parameters that were used in an Rsim scenario

Usage

```
get.rsim.fishing(Rsim.scenario)
```

Arguments

Rsim.scenario Scenario object that contains all of the rsim rates and forcing functions generated by [rsim.scenario\(\)](#).

Value

Returns a ‘fishing’ object.

See Also

Other Get functions: [get.rsim.forcing\(\)](#), [get.rsim.params\(\)](#), [get.rsim.stanzas\(\)](#), [get.rsim.start_state\(\)](#)

Examples

```
# Read in Rpath parameter file and generate balanced model
Rpath <- rpath(AB.params)
# Create a 50 yr Rsim scenario
Rsim.scenario <- rsim.scenario(Rpath, AB.params, years = 1:50)
params <- get.rsim.fishing(Rsim.scenario)
names(params)
```

get.rsim.forcing	<i>Retrieve forcing parameters from an Rsim scenario</i>
------------------	--

Description

Helper function that will retrieve the forcing parameters that were used in an Rsim scenario

Usage

```
get.rsim.forcing(Rsim.scenario)
```

Arguments

`Rsim.scenario` Scenario object that contains all of the rsim rates and forcing functions generated by `rsim.scenario()`.

Value

Returns a ‘forcing’ object.

See Also

Other Get functions: `get.rsim.fishing()`, `get.rsim.params()`, `get.rsim.stanzas()`, `get.rsim.start_state()`

Examples

```
# Read in Rpath parameter file and generate balanced model
Rpath <- rpath(AB.params)
# Create a 50 yr Rsim scenario
Rsim.scenario <- rsim.scenario(Rpath, AB.params, years = 1:50)
params <- get.rsim.forcing(Rsim.scenario)
names(params)
```

get.rsim.params	<i>Retrieve parameters from an Rsim scenario</i>
-----------------	--

Description

Helper function that will retrieve the parameters that were used in an Rsim scenario

Usage

```
get.rsim.params(Rsim.scenario)
```

Arguments

`Rsim.scenario` Scenario object that contains all of the rsim rates and forcing functions generated by `rsim.scenario()`.

Value

Returns a ‘params’ object.

See Also

Other Get functions: `get.rsim.fishing()`, `get.rsim.forcing()`, `get.rsim.stanzas()`, `get.rsim.start_state()`

Examples

```
# Read in Rpath parameter file and generate balanced model
Rpath <- rpath(AB.params)
# Create a 50 yr Rsim scenario
Rsim.scenario <- rsim.scenario(Rpath, AB.params, years = 1:50)
params <- get.rsim.params(Rsim.scenario)
names(params)
```

get.rsim.stanzas	<i>Retrieve stanza parameters from an Rsim scenario</i>
------------------	---

Description

Helper function that will retrieve the stanza parameters that were used in an Rsim scenario

Usage

```
get.rsim.stanzas(Rsim.scenario)
```

Arguments

Rsim.scenario Scenario object that contains all of the rsim rates and forcing functions generated by [rsim.scenario\(\)](#).

Value

Returns a ‘stanzas’ object.

See Also

Other Get functions: [get.rsim.fishing\(\)](#), [get.rsim.forcing\(\)](#), [get.rsim.params\(\)](#), [get.rsim.start_state\(\)](#)

Examples

```
# Read in Rpath parameter file and generate balanced model
Rpath <- rpath(AB.params)
# Create a 50 yr Rsim scenario
Rsim.scenario <- rsim.scenario(Rpath, AB.params, years = 1:50)
params <- get.rsim.stanzas(Rsim.scenario)
names(params)
```

get.rsim.start_state *Retrieve starting state values from an Rsim scenario*

Description

Helper function that will retrieve the starting state values that were used in an Rsim scenario

Usage

```
get.rsim.start_state(Rsim.scenario)
```

Arguments

Rsim.scenario Scenario object that contains all of the rsim rates and forcing functions generated by `rsim.scenario()`.

Value

Returns an 'start_state' object.

See Also

Other Get functions: `get.rsim.fishing()`, `get.rsim.forcing()`, `get.rsim.params()`, `get.rsim.stanzas()`

Examples

```
# Read in Rpath parameter file and generate balanced model
Rpath <- rpath(AB.params)
# Create a 50 yr Rsim scenario
Rsim.scenario <- rsim.scenario(Rpath, AB.params, years = 1:50)
params <- get.rsim.start_state(Rsim.scenario)
names(params)
```

ggwebplot *ggplot routine for Rpath food web*

Description

Plots the food web associated with an Rpath object using ggplot functions

Usage

```
ggwebplot(
  Rpath.obj,
  eco.name = attr(Rpath.obj, "eco.name"),
  line.col = "grey",
  highlight = NULL,
  highlight.col = c("black", "red", "orange"),
  labels = FALSE,
  label.num = FALSE,
  label.cex = 1,
  fleets = FALSE,
  type.col = "grey50",
  box.order = NULL,
  line.alpha = 0.5,
  point.size = 1,
  text.size = 5,
  max.overlaps = 10
)
```

Arguments

<code>Rpath.obj</code>	Rpath model created by the <code>rpath()</code> function.
<code>eco.name</code>	Optional name of the ecosystem. Default is the 'eco.name' attribute from the <code>rpath</code> object.
<code>line.col</code>	The color of the lines between nodes of the food web.
<code>highlight</code>	Set to the group number or name to highlight the connections of that group. Valid values are found in the 'Group' field of the object created from running <code>rpath()</code> .
<code>highlight.col</code>	Color of the connections to the highlighted group, vector of length 3. Defaults to black = predator, red = prey, orange = fleet.
<code>labels</code>	Logical whether or not to display group names.
<code>label.num</code>	Logical whether or not to display group numbers instead of points at nodes. If TRUE, <code>type.col</code> must be length 1, not 4.
<code>label.cex</code>	Numeric value of the relative size of the labels within the plot.
<code>fleets</code>	Logical value indicating whether or not to include fishing fleets in the food web.
<code>type.col</code>	The color of the points corresponding to the types of the group. Can either be of length 1 or 4. Color order will be living, primary producers, detrital, and fleet groups.
<code>box.order</code>	Vector of box numbers to change the default plot order. Must include all box numbers. Passed to <code>summarize.for.webplot()</code>
<code>line.alpha</code>	Transparency of lines between nodes of the food web.
<code>point.size</code>	Size of points at nodes.
<code>text.size</code>	Size of text
<code>max.overlaps</code>	Maximum number of overlaps allowed for group labels by <code>ggrepel</code>

Value

Returns a ggplot object visualizing the food web

Examples

```
# Read in Rpath parameter file, generate and name model object
Rpath.obj <- rpath(AB.params, eco.name = "Anchovy Bay")
# Plot food web diagram with all groups labeled, including fleets, using ggplot
ggwebplot(Rpath.obj, labels = TRUE, fleets = TRUE)
# Plot food web diagram without labels, highlighting connections of cod group
ggwebplot(Rpath.obj, highlight = "cod", fleets = TRUE)
```

import.eiixml

*Reads in EwE exported XML file and parses into data frames***Description**

Parses an eiixml file exported using the Ecopath with Ecosim (EwE) GUI into a list of data frames, one frame for each table in the exported XML file. This function is usually called by the function `create.rpath.from.eiixml()` that converts these tables into an unbalanced rpath model object. However `import.xml` can be used on its own to examine the full set of tables exported by EwE, including tables not currently imported into Rpath objects, such as Ecosim runs or model metadata. This function was tested on files exported by EwE version 6.7.

Usage

```
import.eiixml(eiifile, verbose = FALSE)
```

Arguments

eiifile	Path to exported EwE XML file
verbose	Logical. Use for debugging. If TRUE, prints out a list of parsed tables and number of rows read from the XML file

Value

A list of data frames, one data frame for each node (exported EwE table) in the XML file. Each table has the naming convention `ewe_[table name]` where [table name] is the name of the table provided by EwE.

See Also

Other xml: [create.rpath.from.eiixml\(\)](#)

Examples

```
# Import an eiixml file previously exported from the EwE GUI into a list of
# data frames containing the model data
eiixml <- system.file("extdata/xml", "Western_Bering_Sea.eiixml", package = "Rpath")
xml_data <- import.eiixml(eiixml)
```

MTI

Quantify mixed trophic impacts

Description

Uses a static Rpath model and creates a matrix of mixed trophic impacts.

Usage

```
MTI(Rpath, Rpath.params)
```

Arguments

Rpath	Rpath R object containing a static Rpath model.
Rpath.params	R object containing the Rpath model parameters. This is generated either by the create.rpath.params() or read.rpath.params() functions.

Value

Returns a matrix of mixed trophic impacts.

See Also

Other Rpath functions: [check.rpath.params\(\)](#), [create.rpath.params\(\)](#), [frate.table\(\)](#), [read.rpath.params\(\)](#), [rpath\(\)](#), [rpath.consumers\(\)](#), [rpath.detrital\(\)](#), [rpath.gears\(\)](#), [rpath.groups\(\)](#), [rpath.living\(\)](#), [rpath.mixotrophs\(\)](#), [rpath.producers\(\)](#), [rpath.stanzas\(\)](#), [stanzaplot\(\)](#), [webplot\(\)](#), [write.Rpath\(\)](#), [write.rpath.params\(\)](#)

`read.fitting.biomass` *Read and Process Biomass Fitting Data*

Description

Reads, cleans, and formats biomass observation data to prepare it for model fitting within an 'Rsim' scenario object.

Usage

```
read.fitting.biomass(Rsim.scenario, biomass_data)
```

Arguments

`Rsim.scenario` An 'Rsim' scenario object containing simulation parameters and setup.
`biomass_data` A 'data.frame' containing biomass observation data.

Details

Observation data must be provided in the 'biomass_data' data frame. 'biomass_data' must contain the following columns: 'Group', 'Year', 'Value', 'Scale', 'Source', one of either 'Stdev' or 'CV' (or both), and optionally 'Type'. If both 'Stdev' and 'CV' are provided, 'Stdev' is used for each data point unless it is missing (NA) in which case 'CV' is used.

Value

Modifies and returns a new Rsim scenario object, attaching the cleaned data frame to 'Rsim.scenario\$fitting\$Biomass' with additional calculated columns ('obs', 'sd', 'initial_q', 'wt').

`read.fitting.catch` *Read and Process Catch Fitting Data*

Description

Reads, cleans, and formats catch observation data to prepare it for model fitting within an 'Rpath' simulation scene object.

Usage

```
read.fitting.catch(Rsim.scenario, catch_data)
```

Arguments

`Rsim.scenario` An 'Rsim' scenario object containing simulation parameters and setup.
`catch_data` A 'data.frame' containing catch observation data. Expected to contain the columns: 'Group', 'Year', 'Value', 'Stdev', and 'Scale'.

Details

The function performs several filtering and data-cleaning steps on 'cdat': * Drops species/groups in the data that are not present in the model. * Removes rows containing 'NA' in the 'Value' column. * Filters out observations that fall outside the model's defined run years. * Calculates 'obs' (observed catch) by multiplying 'Value' by 'Scale'. * Calculates 'sd' (standard deviation) by multiplying 'Stdev' by 'Scale'.

Value

Modifies and returns the Rsim.scenario, attaching the formatted data frame to 'Rsim.scenario\$fitting\$Catch' with additional calculated columns ('obs', 'sd', 'wt').

read.rpath.params	<i>Read Rpath parameters from .csv files</i>
-------------------	--

Description

Creates an Rpath.param object from a series of .csv files.

Usage

```
read.rpath.params(
  modfile,
  dietfile,
  pedfile = NA,
  stanzafile = NA,
  stanzafile = NA
)
```

Arguments

modfile	file location of the flat file containing the <i>model</i> parameters.
dietfile	file location of the flat file containing the <i>diet</i> parameters.
pedfile	file location of the flat file containing the <i>pedigree</i> parameters.
stanzagroupfile	file location of the flat file containing the <i>group</i> parameters for multistanza groups. If not specified a blank stanza list will be created.
stanzafile	file location of the flat file containing the individual stanza parameters for multistanza groups. If not specified a blank stanza list will be created.

Value

Outputs an Rpath.param object that can be used for Rpath and subsequently Rsim. (NOTE: This does function does not ensure data is correct or in the correct locations. Run check.rpath.params() to ensure the appropriate columns are present).

See Also

Other Rpath functions: [MTI\(\)](#), [check.rpath.params\(\)](#), [create.rpath.params\(\)](#), [frate.table\(\)](#), [rpath\(\)](#), [rpath.consumers\(\)](#), [rpath.detrital\(\)](#), [rpath.gears\(\)](#), [rpath.groups\(\)](#), [rpath.living\(\)](#), [rpath.mixotrophs\(\)](#), [rpath.producers\(\)](#), [rpath.stanzas\(\)](#), [stanzaplot\(\)](#), [webplot\(\)](#), [write.Rpath\(\)](#), [write.rpath.params\(\)](#)

REco.params

REco.params

Description

Rpath inputs for a tutorial Rpath ecosystem.

Usage

```
REco.params
```

Format

An unbalanced Rpath model object that contains:

model A data.table containing Ecopath unbalanced ecosystem parameters (base parameters and fisheries).

diet A data.table containing the Ecopath model's diet matrix.

stanzas Parameters for multistanza groups.

pedigree A data.table containing the data quality (pedigree) for the Ecopath model. ...

References

Lucey SM, Gaichas SK, Aydin KY (2020) Conducting reproducible ecosystem modeling using the open source mass balance model Rpath. Ecol Model 427:11. doi:10.1016/j.ecolmodel.2020.109057

See Also

`vignette("ModelSetup", package = "Rpath")` for an example of how to build this tutorial ecosystem.

Examples

```
# Balance the model
REco_bal <- rpath(REco.params)
```

rpath	<i>Ecopath module of Rpath</i>
-------	--------------------------------

Description

Performs initial mass balance using a `Rpath.params()` file

Usage

```
rpath(Rpath.params, eco.name = NA, eco.area = 1)
```

Arguments

<code>Rpath.params</code>	R object containing the parameters needed to create a Rpath model. This is generated either by the create.rpath.params() or read.rpath.params() functions.
<code>eco.name</code>	Optional name of the ecosystem which becomes an attribute of the Rpath model.
<code>eco.area</code>	Optional area of the ecosystem which becomes an attribute of the Rpath model.

Value

Returns a static Rpath model that can be supplied to the `rsim.scenario` function.

See Also

Other Rpath functions: [MTI\(\)](#), [check.rpath.params\(\)](#), [create.rpath.params\(\)](#), [frate.table\(\)](#), [read.rpath.params\(\)](#), [rpath.consumers\(\)](#), [rpath.detrital\(\)](#), [rpath.gears\(\)](#), [rpath.groups\(\)](#), [rpath.living\(\)](#), [rpath.mixotrophs\(\)](#), [rpath.producers\(\)](#), [rpath.stanzas\(\)](#), [stanzaplot\(\)](#), [webplot\(\)](#), [write.Rpath\(\)](#), [write.rpath.params\(\)](#)

rpath.consumers	<i>List of Rpath consumer groups</i>
-----------------	--------------------------------------

Description

List of consumer groups from Rpath object with species type = 0.

Usage

```
rpath.consumers(Rpath)
```

Arguments

Rpath	Balanced Rpath model generated by <code>rpath()</code>
-------	--

Value

Returns a string vector containing the names of Rpath functional groups that are consumers

See Also

Other Rpath functions: [MTI\(\)](#), [check.rpath.params\(\)](#), [create.rpath.params\(\)](#), [frate.table\(\)](#), [read.rpath.params\(\)](#), [rpath\(\)](#), [rpath.detrital\(\)](#), [rpath.gears\(\)](#), [rpath.groups\(\)](#), [rpath.living\(\)](#), [rpath.mixotrophs\(\)](#), [rpath.producers\(\)](#), [rpath.stanzas\(\)](#), [stanzaplot\(\)](#), [webplot\(\)](#), [write.Rpath\(\)](#), [write.rpath.params\(\)](#)

Examples

```
# Build the balanced Rpath model and parameter file by calling `rpath`
Rpath <- rpath(Ecosense.EBS)
# Run the consumers command
consumerGroups = rpath.consumers(Rpath)
# Print out the first few consumer group names
head(consumerGroups)
```

rpath.detrital	<i>List of Rpath detrital groups</i>
----------------	--------------------------------------

Description

List of detrital groups from Rpath object with species type = 2

Usage

```
rpath.detrital(Rpath)
```

Arguments

Rpath	Balanced Rpath model generated by <code>rpath()</code>
-------	--

Value

Returns a string vector containing the names of Rpath functional groups that are detrital.

See Also

Other Rpath functions: [MTI\(\)](#), [check.rpath.params\(\)](#), [create.rpath.params\(\)](#), [frate.table\(\)](#), [read.rpath.params\(\)](#), [rpath\(\)](#), [rpath.consumers\(\)](#), [rpath.gears\(\)](#), [rpath.groups\(\)](#), [rpath.living\(\)](#), [rpath.mixotrophs\(\)](#), [rpath.producers\(\)](#), [rpath.stanzas\(\)](#), [stanzaplot\(\)](#), [webplot\(\)](#), [write.Rpath\(\)](#), [write.rpath.params\(\)](#)

Examples

```
# Build the balanced Rpath model and parameter file by calling `rpath`
Rpath <- rpath(Ecosense.EBS)
# Run the rpath.detrital command
detritalGroups = rpath.detrital(Rpath)
# Print out the first few detrital group names
head(detritalGroups)
```

rpath.gears

List of Rpath gears groups

Description

List of gears groups from Rpath object with species type = 3

Usage

```
rpath.gears(Rpath)
```

Arguments

Rpath Balanced Rpath model generated by rpath()

Value

Returns a string vector containing the names of Rpath functional groups that are gear types

See Also

Other Rpath functions: [MTI\(\)](#), [check.rpath.params\(\)](#), [create.rpath.params\(\)](#), [frate.table\(\)](#), [read.rpath.params\(\)](#), [rpath\(\)](#), [rpath.consumers\(\)](#), [rpath.detrital\(\)](#), [rpath.groups\(\)](#), [rpath.living\(\)](#), [rpath.mixotrophs\(\)](#), [rpath.producers\(\)](#), [rpath.stanzas\(\)](#), [stanzaplot\(\)](#), [webplot\(\)](#), [write.Rpath\(\)](#), [write.rpath.params\(\)](#)

Examples

```
# Build the balanced Rpath model and parameter file by calling `rpath`
Rpath <- rpath(Ecosense.EBS)
# Run the groups command
gearGroups = rpath.gears(Rpath)
# Print out the first few gear type group names
head(gearGroups)
```

rpath.groups	<i>Rpath functional group names</i>
--------------	-------------------------------------

Description

Get a character vector of functional group names from an Rpath object (balanced model) or an Rpath.params object (unbalanced model parameters) based on the 'type' input parameter as follows: (0: consumers, 1: producers, 2: detrital, 3: gears, 0<type<1: mixotrophs). Living groups are consumers + producers. Note that mixotrophs are not returned as either consumers or producers, only separately.

Usage

```
rpath.groups(Rpath)
```

Arguments

Rpath	Balanced Rpath model generated by rpath()
-------	---

Value

Returns a string vector containing the names of Rpath functional groups by category (group type).

See Also

Other Rpath functions: [MTI\(\)](#), [check.rpath.params\(\)](#), [create.rpath.params\(\)](#), [frate.table\(\)](#), [read.rpath.params\(\)](#), [rpath\(\)](#), [rpath.consumers\(\)](#), [rpath.detrital\(\)](#), [rpath.gears\(\)](#), [rpath.living\(\)](#), [rpath.mixotrophs\(\)](#), [rpath.producers\(\)](#), [rpath.stanzas\(\)](#), [stanzaplot\(\)](#), [webplot\(\)](#), [write.Rpath\(\)](#), [write.rpath.params\(\)](#)

Examples

```
# Build the balanced Rpath model and parameter file by calling `rpath`
Rpath <- rpath(Ecosense.EBS)
# Run the groups command
groups = rpath.groups(Rpath)
# Print out the first few group names
head(groups)
```

rpath.living	<i>List of Rpath living groups</i>
--------------	------------------------------------

Description

List of living groups from Rpath object. Living groups are those species groups with type < 2.

Usage

```
rpath.living(Rpath)
```

Arguments

Rpath	Balanced Rpath model generated by rpath()
-------	---

Value

Returns a string vector containing the names of Rpath functional groups that are living.

See Also

Other Rpath functions: [MTI\(\)](#), [check.rpath.params\(\)](#), [create.rpath.params\(\)](#), [frate.table\(\)](#), [read.rpath.params\(\)](#), [rpath\(\)](#), [rpath.consumers\(\)](#), [rpath.detrital\(\)](#), [rpath.gears\(\)](#), [rpath.groups\(\)](#), [rpath.mixotrophs\(\)](#), [rpath.producers\(\)](#), [rpath.stanzas\(\)](#), [stanzaplot\(\)](#), [webplot\(\)](#), [write.Rpath\(\)](#), [write.rpath.params\(\)](#)

Examples

```
# Build the balanced Rpath model and parameter file by calling `rpath`
Rpath <- rpath(Ecosense.EBS)
# Run the rpath.living command
livingGroups = rpath.living(Rpath)
# Print out the first few living group names
head(livingGroups)
```

rpath.mixotrophs	<i>List of Rpath mixotroph groups</i>
------------------	---------------------------------------

Description

List of groups from Rpath object with species type between 0 and 1

Usage

```
rpath.mixotrophs(Rpath)
```

Arguments

Rpath Balanced Rpath model generated by `rpath()`

Value

Returns a string vector containing the names of Rpath functional groups that are mixotrophs

See Also

Other Rpath functions: `MTI()`, `check.rpath.params()`, `create.rpath.params()`, `frate.table()`, `read.rpath.params()`, `rpath()`, `rpath.consumers()`, `rpath.detrital()`, `rpath.gears()`, `rpath.groups()`, `rpath.living()`, `rpath.producers()`, `rpath.stanzas()`, `stanzaplot()`, `webplot()`, `write.Rpath()`, `write.rpath.params()`

Examples

```
# Build the balanced Rpath model and parameter file by calling `rpath`
Rpath <- rpath(Ecosense.EBS)
# Run the groups command
mixotrophGroups = rpath.mixotrophs(Rpath)
# Print out the first few mixotroph group names
head(mixotrophGroups)
```

rpath.producers	<i>List of Rpath producer groups</i>
-----------------	--------------------------------------

Description

List of producer groups from Rpath object with species type = 1

Usage

```
rpath.producers(Rpath)
```

Arguments

Rpath Balanced Rpath model generated by `rpath()`

Value

Returns a string vector containing the names of Rpath functional groups that are producers

See Also

Other Rpath functions: `MTI()`, `check.rpath.params()`, `create.rpath.params()`, `frate.table()`, `read.rpath.params()`, `rpath()`, `rpath.consumers()`, `rpath.detrital()`, `rpath.gears()`, `rpath.groups()`, `rpath.living()`, `rpath.mixotrophs()`, `rpath.stanzas()`, `stanzaplot()`, `webplot()`, `write.Rpath()`, `write.rpath.params()`

Examples

```
# Build the balanced Rpath model and parameter file by calling `rpath`
Rpath <- rpath(Ecosense.EBS)
# Run the groups command
producerGroups = rpath.producers(Rpath)
# Print out the first few producer group names
head(producerGroups)
```

rpath.stanzas

Calculate biomass and consumption for multistanza groups

Description

Uses the leading stanza to calculate the biomass and consumption of other stanzas necessary to support the leading stanza.

Usage

```
rpath.stanzas(Rpath.params)
```

Arguments

Rpath.params R object containing the parameters needed to create a Rpath model. This is generated either by the [create.rpath.params\(\)](#) or [read.rpath.params\(\)](#) functions.

Value

Calculates and adds biomass and consumption for trailing stanza groups. Also adds weight at age and number at age for multi-staza groups.

See Also

Other Rpath functions: [MTI\(\)](#), [check.rpath.params\(\)](#), [create.rpath.params\(\)](#), [frate.table\(\)](#), [read.rpath.params\(\)](#), [rpath\(\)](#), [rpath.consumers\(\)](#), [rpath.detrital\(\)](#), [rpath.gears\(\)](#), [rpath.groups\(\)](#), [rpath.living\(\)](#), [rpath.mixotrophs\(\)](#), [rpath.producers\(\)](#), [stanzaplot\(\)](#), [webplot\(\)](#), [write.Rpath\(\)](#), [write.rpath.params\(\)](#)

rsim.deriv	<i>Calculate the derivatives for a time step</i>
------------	--

Description

Calculates the derivative for a single time step and saves the output

Usage

```
rsim.deriv(Rsim.scenario, sim.year = 0, sim.month = 0, timestep = 0, outlist = F)
```

Arguments

Rsim.scenario	Scenario object that contains all of the rsim rates and forcing functions generated by <code>rsim.scenario()</code> .
sim.year	Year of fishing/forcing timeseries to use for derivative calculations.
sim.month	Month of fishing/forcing timeseries to use for derivative calculations, noting that month uses a (0-11) range rather than (1-12) due to C++ indexing offsets.
timestep	Sub-monthly time step usually set to 0.
outlist	(optional) return output as unlabeled list instead of labeled table.

Value

If `outlist` is true, returns the raw unlabeled derivative list used by the Rcpp integration code, otherwise returns a labeled data.frame object of model group (rows) values for 12 derivatives (columns):

- Species, character names of model groups
- DerivT, numeric net flow; Total Gain - Total Loss at the timestep
- TotGain, numeric Total Gain, all flows into the group at the timestep
- TotLoss, numeric Total Loss, all flows out of the group at the timestep
- FoodGain, numeric flows into the group from consuming prey at the timestep
- DetritalGain, numeric flows into the group from detritus at the timestep
- FishingGain, numeric flows into the group from fishing at the timestep
- UnAssimLoss, numeric flows out of the group due to unassimilated consumption at the timestep
- ActiveRespLoss, numeric flows out of the group due to "heat loss" at the timestep
- FoodLoss, numeric flows out of the group from being consumed by predators at the timestep
- MzeroLoss, numeric flows out of the group due to unaccounted mortality at the timestep
- FishingLoss, numeric flows out of the group due to fishing at the timestep
- DetritalLoss, numeric flows out of the group to detritus at the timestep

Examples

```
# Read in Rpath parameter file and generate model object
Rpath <- rpath(AB.params)
# Create a 50 yr Rsim scenario
Rsim.scenario <- rsim.scenario(Rpath, AB.params, years = 1:50)
# Calculate derivatives for year 2, month 6, first timestep
Rsim.deriv <- rsim.deriv(Rsim.scenario, sim.year=2, sim.month = 6, timestep = 0)
```

rsim.diet	<i>Output consumption of each prey by an individual predator group from Rsim run</i>
-----------	--

Description

Extracts a matrix of consumption (in model biomass units) of each prey over time by a particular predator from the Rsim model output of `rsim.run()`.

Usage

```
rsim.diet(Rsim.output, group)
```

Arguments

Rsim.output	R object containing the output from <code>rsim.run</code> .
group	Predator group from the Rpath model that is of interest

Value

a numeric matrix of annual (rows) consumption of each prey group (columns)

Examples

```
# Read in Rpath parameter file and generate model object
Rpath <- rpath(AB.params)
# Create a 50 yr Rsim scenario
Rsim.scenario <- rsim.scenario(Rpath, AB.params, years = 1:50)
# Run the Rsim simulation
Rsim.output <- rsim.run(Rsim.scenario, method = "RK4", years = 1:50)
# Extract a predator's consumption of each prey over the model run
Rsim.diet.whales <- rsim.diet(Rsim.output, group = "whales")
```

rsim.fishing	<i>Initialize fishing matrix for Rsim scenario</i>
--------------	--

Description

Creates a matrix for forcing functions related to fishing: (*ForcedEffort*, *ForcedFRate*, and *ForcedCatch*). Default values are set to 1 for *ForcedEffort*, and to 0 *ForcedFRate*, and *ForcedCatch* for all timesteps and groups.

Usage

```
rsim.fishing(params, years)
```

Arguments

params	Rsim parameter object generated by rsim.params.
years	A vector of each year of the simulation.

Value

S3 object class Rsim.fishing, a list of 3 objects:

- *ForcedEffort*, numeric matrix of monthly (rows) effort multiplier by fleet (columns)
- *ForcedFRate*, numeric matrix of annual (rows) fishing mortality rate by biomass group (columns)
- *ForcedEffort*, numeric matrix of annual (rows) total catch removal by biomass group (columns)

Examples

```
# Read in Rpath parameter file and generate model object
Rpath <- rpath(AB.params)
# Create dynamic parameters from Rpath model
Rsim.params <- rsim.params(Rpath)
# Create fishing matrix with default values
Rsim.fishing <- rsim.fishing(Rsim.params, years = 1:50)
```

rsim.fit.apply	<i>Apply Fitted Parameters to Rpath Simulation Scene Helper function that maps a vector of optimized parameter modifications back into the 'Rpath' simulation parameters object ('scene.params'). Since this returns a scenario params object instead of a full scenario, it is primarily intended for internal use.</i>
----------------	--

Description

Apply Fitted Parameters to Rpath Simulation Scene Helper function that maps a vector of optimized parameter modifications back into the 'Rpath' simulation parameters object ('scene.params'). Since this returns a scenario params object instead of a full scenario, it is primarily intended for internal use.

Usage

```
rsim.fit.apply(values, species, vartype, scene.params)
```

Arguments

values	A numeric vector of parameter values or modifiers. For a parameter of var-type "mzero", the unit of the value is a mortality rate. Values for "predvul", "preyvul", and "ppvul" are log-scaled vulnerabilities.
species	A character vector indicating the species/groups to modify (or string representations of numeric link indices for "ppvul").
vartype	A character vector indicating the types of parameters to modify. Currently supported parameters are "mzero", "predvul", "preyvul", and "ppvul".
scene.params	An 'Rpath' simulation parameters list (usually 'scene\$params').

Details

This function processes a vector of parameter modifications ('values') based on their assigned 'var-type'. It handles four parameter types: * "mzero": Modifies non-predation mortality directly. * "predvul": Modifies vulnerability from the predator's perspective. * "preyvul": Modifies vulnerability from the prey's perspective. * "ppvul": Modifies vulnerability for a single, specific predator-prey link. For vulnerabilities, the modifications are applied to the base vulnerability ('VV') using a logarithmic transformation: $VV_{new} = 1 + \exp(\log(VV_{base} - 1) + \text{modifications})$.

Value

Modifies and returns the 'scene.params' object with updated 'MzeroMort' and 'VV' matrices.

`rsim.fit.list.bio.bysurvey`*List Biomass Timeseries Groups by Survey Source*

Description

Extracts and groups the unique species or functional groups associated with each survey source in the biomass fitting data of an ‘Rsim’ simulation scene.

Usage

```
rsim.fit.list.bio.bysurvey(Rsim.scenario)
```

Arguments

`Rsim.scenario` An ‘Rpath’ simulation scene list object containing the fitting data setup.

Details

This helper function reads the ‘Source’ and ‘Group’ columns from the ‘scene\$fitting\$Biomass’ data frame. It iterates through every unique survey source and compiles a list of all groups (species) that have observation data attributed to that specific source.

Value

A named list where each element name corresponds to a unique survey ‘Source’, and the contents are character vectors of the ‘Group’ names associated with that source.

`rsim.fit.list.bio.series`*List Biomass Timeseries Groups and Sources*

Description

Extracts all unique groups, survey sources, and combined identifiers from the biomass fitting data within an ‘Rsim’ simulation scene.

Usage

```
rsim.fit.list.bio.series(Rsim.scenario)
```

Arguments

`Rsim.scenario` An ‘Rsim’ simulation scene list object containing the fitting data setup.

Details

This helper function reads the ‘Group’ and ‘Source’ columns from the ‘scene\$fitting\$Biomass’ data frame. It returns a summary list containing the unique functional groups, the unique survey sources, and the unique combinations of the two formatted as “Group:Source”.

Value

A list containing three elements: * ‘groups’: A character vector of unique species/groups. * ‘sources’: A character vector of unique survey sources. * ‘all’: A character vector of unique combinations (“Group:Source”).

```
rsim.fit.list.catch.series
```

List Catch Timeseries Groups

Description

Extracts the unique species or functional groups that have observation data within the catch fitting matrix of an ‘Rsim’ simulation scenario.

Usage

```
rsim.fit.list.catch.series(Rsim.scenario)
```

Arguments

`Rsim.scenario` An ‘Rsim’ scenario list object containing fitting data setup.

Details

This is a helper function designed to quickly identify which groups have empirical catch data loaded into the scenario. It reads the ‘Group’ column from the ‘Rsim.scenario\$fitting\$Catch’ data frame and returns the unique values.

Value

A list containing a single element, ‘groups’, which holds a vector of the unique group names present in the catch data.

rsim.fit.obj

*Calculate Objective Function for Rsim Model Fit***Description**

Computes the negative log-likelihood (goodness of fit) between an 'Rsim' simulation's dynamic output and empirical observation timeseries data for both Biomass and Catch.

Usage

```
rsim.fit.obj(Rsim.scenario, Rsim.output, verbose = TRUE)
```

Arguments

Rsim.scenario	An 'Rsim' scenario object containing the fitting data.
Rsim.output	An Rsim simulation output to be evaluated.
verbose	Logical; if 'TRUE', returns a detailed list containing the original fitting data appended with model estimates, 'q' values, and individual fit scores. If 'FALSE', returns only the total sum of the negative log-likelihoods (useful for passing to optimizer functions).

Details

This function evaluates how well a simulation run ('Rsim.output') matches the loaded observation data ('Rsim.scenario\$fitting'). It uses a log-normal error distribution and calculates the negative log-likelihood.

For biomass observation series marked as '"index"', the function analytically calculates a variance-weighted catchability coefficient ('q_est') to scale the estimates prior to calculating the log-likelihood error.

Value

If 'verbose = TRUE', returns a list containing detailed data frames ('Biomass', 'Catch') and the total objective score ('tot'). If 'verbose = FALSE', returns a single numeric scalar representing the total negative log-likelihood.

rsim.fit.obj.species

*Extract Objective Function Fits by Species***Description**

Runs the objective function and extracts the detailed, row-by-row fit scores strictly for specific species or functional groups from both the Biomass and Catch data.

Usage

```
rsim.fit.obj.species(Rsim.scenario, Rsim.output, species = NULL)
```

Arguments

Rsim.scenario An 'Rsim' scenario object containing the fitting data.

Rsim.output An Rsim simulation output to be evaluated.

species A character vector of species or group names to extract. Defaults to 'NULL'.

Details

This function acts as a filter on top of 'rsim.fit.obj()'. It runs the base objective function in verbose mode to calculate the individual negative log-likelihoods for all observation data. It then subsets the resulting 'Biomass' and 'Catch' data frames, returning only the rows where the 'Group' matches the provided 'species' vector.

Value

A list containing two data frames ('Biomass' and 'Catch') representing the detailed observation-level fit data filtered to the requested species.

```
rsim.fit.remove.bio.timeseries
```

Remove Biomass Timeseries from Fitting Data

Description

Removes specific biomass observation timeseries from an 'Rsim' scenario object based on their unique identification strings.

Usage

```
rsim.fit.remove.bio.timeseries(Rsim.scenario, timeseries)
```

Arguments

Rsim.scenario An 'Rsim' scenario object containing the fitting data setup.

timeseries A character vector of 'ID' strings representing the timeseries to be removed (e.g., "SpeciesName:SurveySource"). Existing timeseries IDs can be listed using the 'rsim.fit.list.bio.series' function.

Details

This is a utility function used to drop specific observation datasets (e.g., a specific survey for a specific species) prior to running a model fit. It filters the 'scene\$fitting\$Biomass' data frame, keeping only the rows where the 'ID' does NOT match the values provided in the 'timeseries' argument.

Value

Modifies and returns the 'Rsim.scenario' object, with the specified timeseries removed from the 'Rsim.scenario\$fitting\$Biomass' data frame.

rsim.fit.run

*Run Fitted Rpath Simulation and Evaluate Objective Function***Description**

Applies parameter modifications to an 'Rpath' scene, runs the simulation, and either returns the full simulation output or evaluates the fit against observation data using a penalized negative log-likelihood.

Usage

```
rsim.fit.run(
  values,
  species,
  vartype,
  Rsim.scenario,
  run_method,
  run_years,
  verbose = F,
  penalty_weight = 0,
  nll_details = FALSE,
  ...
)
```

Arguments

values	A numeric vector of parameter values or modifiers. For a parameter of var-type "mzero", the unit of the value is a mortality rate. Values for "predvul", "preyvul", and "ppvul" are log-scaled vulnerabilities.
species	A character vector indicating the species/groups to modify (or string representations of numeric link indices for "ppvul").
vartype	A character vector indicating the types of parameters to modify. Currently supported parmeters are "mzero", "predvul", "preyvul", and "ppvul".
Rsim.scenario	An 'Rsim' scenario object containing the fitting data.
run_method	Numerical integration method for rsim.run. Either 'AB' for Adams-Bashforth or 'RK4' for 4th order Runge-Kutta.
run_years	A numeric vector of years over which to run the simulation.
verbose	Logical; if 'FALSE', returns the NLL fit score. If 'TRUE', returns the complete simulation run object. Defaults to 'F'.
penalty_weight	Numeric value to scale the L2 penalty term for parameter regularization. Defaults to '0'.

`nll_details` Logical; if 'TRUE' and 'verbose = FALSE', returns a list breaking down the total NLL into base error and penalty components.

... Additional arguments (currently swallowed and unused by the function).

Details

This function translates the defined fitting variables into the appropriate variables of an `Rsim.scenario` object and runs an `rsim` simulation with the resulting scenario. Applying an "mzero" parameter to a species replaces the `Rsim.scenario$params$MzeroMort` paramter for that species. "predvul", "preyvul", and "ppvul" are applied using the formula for each predator/prey link `Rsim.scenario$params$VV(predator, prey) = (1 + exp(log(Rsim.scenario$params$VV(predator, prey)-1) + preddiff + preydifff + ppdiff))`

Value

* If 'verbose = TRUE': Returns an 'Rsim' simulation run output object. * If 'verbose = FALSE' and 'nll_details = FALSE': Returns a single numeric value representing the total penalized negative log-likelihood. * If 'verbose = FALSE' and 'nll_details = TRUE': Returns a named list containing 'total_nll', 'base_error', and 'penalty_term'.

`rsim.fit.set.bio.wt` *Set Biomass Fitting Data Weights*

Description

Updates the statistical weighting ('wt') for specific biomass observation timeseries prior to running a model fit in an 'Rsim' simulation scene. Matches are based on both the functional group and the survey source.

Usage

```
rsim.fit.set.bio.wt(Rsim.scenario, group, source, wt)
```

Arguments

`Rsim.scenario` An `Rsim` scenario object containing the fitting data.

`group` A character string or vector of group names to update.

`source` A character string or vector of survey source names to update.

`wt` A numeric value to assign as the new weight.

Details

This function locates all observation rows in the `'Rsim.scenario$fitting$Biomass'` data frame that match BOTH the specified 'group' and 'source' names. It replaces their "wt" column with the provided 'wt' value. If the specific combination is not found in the biomass data, it issues a warning and returns the scene unmodified.

Value

Modifies and returns the 'Rsim.scenario' object with the updated weights applied to 'Rsim.scenario\$fitting\$Biomass'.

rsim.fit.set.catch.wt *Set Catch Fitting Data Weights*

Description

Updates the statistical weighting ('wt') for specific catch observation timeseries prior to running a model fit in an 'Rsim' simulation scene. Matches are based on both the functional group and the survey source.

Usage

```
rsim.fit.set.catch.wt(Rsim.scenario, group, wt)
```

Arguments

Rsim.scenario	An 'Rsim' scenario object containing the fitting data.
group	A character string or vector of group names to update.
wt	A numeric value to assign as the new weight.

Details

This function locates all observation rows in the 'Rsim.scenario\$fitting\$Catch' data frame that match BOTH the specified 'group' and 'source' names. It replaces their "wt" column with the provided 'wt' value. If the specific combination is not found in the catch data, it issues a warning and returns the scene unmodified.

Value

Modifies and returns the 'Rsim.scenario' object with the updated weights applied to 'Rsim.scenario\$fitting\$Biomass'.

rsim.fit.set.q *Set Catchability (q) for Biomass Fitting Data*

Description

Updates the catchability coefficient ('initial_q') and timeseries 'Type' for specific biomass observations within an 'Rpath' simulation scene.

Usage

```
rsim.fit.set.q(
  Rsim.scenario,
  group,
  source,
  q = NULL,
  years = NULL,
  type = NULL
)
```

Arguments

Rsim.scenario	An Rsim scenario object containing the fitting data.
group	A character string (or vector) of the functional group(s) to update.
source	A character string (or vector) of the survey source(s) to update.
q	An optional numeric value to directly set the catchability coefficient.
years	An optional numeric or character vector of years used to calculate ‘q’ empirically.
type	An optional character string to set the timeseries type (defaults to “fixed”).

Details

This function operates in three distinct modes depending on the arguments provided: 1. **Default Reset** (‘q’ and ‘years’ are ‘NULL’): Resets ‘initial_q’ to ‘1.0’ and sets ‘Type’ to “index”. 2. **Direct Assignment** (‘q’ is provided): Assigns the provided numeric ‘q’ to ‘initial_q’ and assigns the specified ‘type’ (which defaults to “fixed”). 3. **Empirical Calculation** (‘years’ is provided, ‘q’ is ‘NULL’): Calculates ‘q’ by taking the mean observed biomass (‘Value * Scale’) over the specified ‘years’ and dividing it by the model’s base reference biomass (‘B_BaseRef’) for the specified ‘group’.

Value

Modifies and returns the ‘Rsim.scenario’ (scene) object with updated ‘initial_q’ and ‘Type’ columns in the ‘scene\$fitting\$Biomass’ data frame.

rsim.fit.table

Extract Summary Table of Model Fits

Description

Runs the objective function and aggregates the negative log-likelihood fit scores by functional group for both Biomass and Catch data, returning them in a standardized data frame.

Usage

```
rsim.fit.table(Rsim.scenario, Rsim.output)
```

Arguments

- `Rsim.scenario` An Rsim scenario object containing the fitting data.
- `Rsim.output` An Rsim simulation output to be evaluated.

Details

This function acts as a convenient wrapper around `'rsim.fit.obj()'`. It runs the objective function in verbose mode to extract individual observation fit scores, sums those scores by species/group, and then aligns the results against the complete species list defined in the model.

Value

A `'data.frame'` where the row names correspond to the model's full species list. It contains two columns (`'Biomass'` and `'Catch'`) representing the summed negative log-likelihood fit scores for each group.

<code>rsim.fit.update</code>	<i>Update Rpath Scene with Fitted Parameters</i>
------------------------------	--

Description

A function that applies fit variables to an Rsim scenario, returning a new scenario object.

Usage

```
rsim.fit.update(values, species, vartype, Rsim.scenario)
```

Arguments

- `values` A numeric vector of parameter values or modifiers. For a parameter of vartype `"mzero"`, the unit of the value is a mortality rate. Values for `"predvul"`, `"preyvul"`, and `"ppvul"` are log-scaled vulnerabilities.
- `species` A character vector indicating the species/groups to modify (or string representations of numeric link indices for `"ppvul"`).
- `vartype` A character vector indicating the types of parameters to modify. Currently supported parameters are `"mzero"`, `"predvul"`, `"preyvul"`, and `"ppvul"`.
- `Rsim.scenario` An `'Rsim'` scenario object containing the fitting data.

Details

This function translates the defined fitting variables into the appropriate variables of an `Rsim.scenario` object. In particular, applying an `"mzero"` parameter to a species replaces the `Rsim.scenario$params$MzeroMort` parameter for that species. `"predvul"`, `"preyvul"`, and `"ppvul"` are applied using the formula for each predator/prey link $Rsim.scenario$params$VV(predator, prey) = (1 + \exp(\log(Rsim.scenario$params$VV(predator, prey) - 1) + prediff + preydiff + ppdiff))$

Value

Modifies and returns the complete 'Rsim.scenario' object with updated vulnerability and mortality parameters.

rsim.forcing	<i>Initialize forcing matrix for Rsim scenario</i>
--------------	--

Description

Creates a matrix for forcing functions not related to fishing (*ForcedPrey ForcedMort, ForcedRecs, ForcedSearch, ForcedActResp, ForcedMigrate, and ForcedBio*). Default values are set to 1 for *ForcedPrey ForcedMort, ForcedRecs, ForcedSearch* and *ForcedActResp*, to 0 for *ForcedMigrate* and to -1 for *ForcedBio* for all timesteps and groups.

Usage

```
rsim.forcing(params, years)
```

Arguments

params	Rsim parameter object generated by rsim.params.
years	A vector of each year of the simulation.

Value

S3 object class Rsim.forcing, a list of 7 objects:

- ForcedPrey, numeric matrix of monthly (rows) prey forcing by biomass group (columns)
- ForcedMort, numeric matrix of monthly (rows) mortality multiplier by biomass group (columns)
- ForcedRecs, numeric matrix of monthly (rows) recruitment multiplier by biomass group (columns)
- ForcedSearch, numeric matrix of monthly (rows) search rate multiplier by biomass group (columns)
- ForcedActresp, numeric matrix of monthly (rows) active respiration multiplier by biomass group (columns)
- ForcedMigrate, numeric matrix of monthly (rows) migration forcing by biomass group (columns)
- ForcedBio, numeric matrix of monthly (rows) biomass multiplier by biomass group (columns)

Examples

```
# Read in Rpath parameter file and generate model object
Rpath <- rpath(AB.params)
# Create dynamic parameters from Rpath model
Rsim.params <- rsim.params(Rpath)
# Create forcing matrix with default values
Rsim.forcing <- rsim.forcing(Rsim.params, years = 1:50)
```

rsim.mort	<i>Output biomass removed by each predator for an individual prey group from Rsim run</i>
-----------	---

Description

Extracts a matrix of biomass removed (in model biomass units) for a particular prey by each of its predators from the Rsim model output of `rsim.run()`.

Usage

```
rsim.mort(Rsim.output, group)
```

Arguments

Rsim.output	R object containing the output from <code>rsim.run</code> .
group	Predator group from the Rpath model that is of interest

Value

a numeric matrix of annual (rows) biomass removed by each predator group (columns)

Examples

```
# Read in Rpath parameter file and generate model object
Rpath <- rpath(AB.params)
# Create a 50 yr Rsim scenario
Rsim.scenario <- rsim.scenario(Rpath, AB.params, years = 1:50)
# Run the Rsim simulation
Rsim.output <- rsim.run(Rsim.scenario, method = "RK4", years = 1:50)
# Extract a prey's biomass loss from each predator over the model run
Rsim.mort.anchovy <- rsim.mort(Rsim.output, group = "anchovy")
```

rsim.params	<i>Initial set up for Rsim dynamic simulation</i>
-------------	---

Description

Converts the outputs from Rpath into rates for use in Rsim.

Usage

```
rsim.params(
  Rpath,
  mscramble = 2,
  mhandle = 1000,
  preyswitch = 1,
  scrambleselfwt = 0,
  handleselfwt = 0,
  steps_yr = 12,
  steps_m = 1
)
```

Arguments

Rpath	R object containing a static Rpath model.
mscramble	Base value for vulnerability in functional response; default = 2 (mixed response).
mhandle	Base value for handling time in functional response; default = 1000 (off).
preyswitch	Exponent for prey switching in functional response; default = 1 (off).
scrambleselfwt	Value of 1 indicates all predators overlap in the foraging arena while 0 treats predators individually.
handleselfwt	Value of 1 indicates all prey overlap in the arena and contribute to saturation while 0 treats prey individually.
steps_yr	Number of time steps per year; default = 12.
steps_m	Number of time steps per month; default = 1.

Value

Returns an object of class `Rsim.params`, a list of 39 objects that is passed to the `rsim.run` function via the `rsim.scenario` function.

- NUM_GROUPS, number of total model groups
- NUM_LIVING, number of living model groups
- NUM_DEAD, number of detritus model groups
- NUM_GEARS, number of fishery model groups
- NUM_BIO, number of living + detritus model groups
- spname, named NUM_GROUPS+1 length character vector beginning with "Outside" containing names of all model groups
- spnum, named NUM_GROUPS+1 length character vector beginning with "Outside" numbered 0, containing numbers of all model groups
- B_BaseRef, named NUM_GROUPS+1 length character vector beginning with "Outside" containing Rpath base biomass of all model groups
- MzeroMort, named NUM_GROUPS+1 length character vector beginning with "Outside" containing unaccounted mortality, calculated as $PB * (1.0 - EE)$, of all model groups

- UnassimRespFrac, named NUM_GROUPS+1 length character vector beginning with "Outside" containing the proportion of consumption that goes to detritus of all model groups
- ActiveRespFrac, named NUM_GROUPS+1 length character vector beginning with "Outside" containing the proportion of consumption that is "lost to heat" for all model groups
- FtimeAdj, named NUM_GROUPS+1 length character vector beginning with "Outside" containing rate of change of feeding time, currently set to 0 for all model groups
- FtimeQBOpt, named NUM_GROUPS+1 length character vector beginning with "Outside" containing base QB for all consumer model groups, or base PB for primary producers
- PBopt, named NUM_GROUPS+1 length character vector beginning with "Outside" containing base PB for all model groups
- NoIntegrate, named NUM_GROUPS+1 length character vector beginning with "Outside" containing flag set to 0 for high turnover model groups and set to spnum for all others
- Handleself, named NUM_GROUPS+1 length character vector beginning with "Outside" containing flag for handling time influence, set for all model groups with function argument handleselfwt, default 0 for individual prey handling time
- Scrambleself, named NUM_GROUPS+1 length character vector beginning with "Outside" containing flag for predator density influence, set for all model groups with function argument scrambleselfwt, default 0 for individual predator density dependent predation
- PreyFrom, numeric vector length NumPredPreyLinks+1, spnum of prey for each predator prey interaction pair in the model
- PreyTo, numeric vector length NumPredPreyLinks+1, spnum of predator for each predator prey interaction pair in the model
- QQ, numeric vector length NumPredPreyLinks+1, base consumption rate for each predator prey interaction pair in the model
- DD, numeric vector length NumPredPreyLinks+1, handling time effect on functional response for each predator prey pair, set for all model groups with function argument mhandle, default = 1000
- VV, numeric vector length NumPredPreyLinks+1, vulnerability effect on functional response for each predator prey pair, set for all model groups with function argument mscramble, default = 2
- HandleSwitch, numeric vector length NumPredPreyLinks+1, prey density dependence effect on functional response for each predator prey pair, set for all model groups with function argument preyswitch, default = 1
- PredPreyWeight, numeric vector length NumPredPreyLinks+1, relative weight of individual predator to total predators for each predator prey pair, used if scrambleselfwt>0
- PreyPreyWeight, numeric vector length NumPredPreyLinks+1, relative weight of individual prey to total prey for each predator prey pair, used if handleselfwt>0
- NumPredPreyLinks, number of predator to prey linkages over all groups in the model
- FishFrom, numeric vector length NumFishingLinks+1, spnum of landing and discard for each fishery interaction in the model
- FishThrough, numeric vector length NumFishingLinks+1, spnum of gear type for each fishery interaction in the model

- FishQ, numeric vector length NumFishingLinks+1, landings or discards relative to base fished group biomass for each fishery interaction in the model
- FishTo, numeric vector length NumFishingLinks+1, spnum of sink for each fishery interaction in the model ("Outside" for landings or detritus group for discards)
- NumFishingLinks, number of model group landings and discards to fishery links over all groups in the model
- DetFrac, numeric vector length NumDetLinks+1, fraction of detritus going to DetTo for each living and detritus group in the model
- DetFrom, numeric vector length NumDetLinks+1, spnum flowing to detritus for each living and detritus group in the model
- DetTo, numeric vector length NumDetLinks+1, spnum of detritus sink for each living and detritus group in the model
- NumDetLinks, number of model group links to detritus over all groups in the model
- BURN_YEARS, number of model run burn-in (spin up) years, default value -1
- COUPLED, number to control species interactions, value of 0 allows density dependent non-interacting species, default value 1
- RK4_STEPS, number of RK4 integration steps per month, default value 4
- SENSE_LIMIT, numeric vector of multipliers on biomass determining acceptable range for continuing a model run within BURN_YEARS

See Also

Other Rsim functions: [extract.node\(\)](#), [rsim.plot\(\)](#), [rsim.run\(\)](#), [rsim.scenario\(\)](#), [rsim.step\(\)](#), [write.Rsim\(\)](#)

Examples

```
# Read in Rpath parameter file and generate model object
Rpath <- rpath(AB.params)
# Create default dynamic parameters from Rpath model
Rsim.params <- rsim.params(Rpath)
```

rsim.plot

Plot routine for Rsim runs

Description

Plots the relative biomass of each group from an Rsim.output object.

Usage

```
rsim.plot(Rsim.output, spname = "all", indplot = F)
```

Arguments

Rsim.output	R object containing the output from <code>rsim.run()</code> .
spname	Vector of species names to be displayed. The default "all" will display all group names
indplot	Logical value of whether to plot a single group or multiple groups

Value

Creates a figure of relative biomass

Details

For a detailed example of how to use this plot command please see `vignette("RunRsim")`

See Also

Other Rsim functions: `extract.node()`, `rsim.params()`, `rsim.run()`, `rsim.scenario()`, `rsim.step()`, `write.Rsim()`

rsim.plot.fit	<i>Plot Biomass and Catch Fitting Results</i>
---------------	---

Description

Generates multi-panel diagnostic plots comparing simulation run outputs against observed biomass and catch data across a specified list of species.

Usage

```
rsim.plot.fit(Rsim.scenario, Rsim.output, species.list, scene.name = NULL)
```

Arguments

Rsim.scenario	An 'Rsim' scenario object containing the fitting data.
Rsim.output	An 'Rsim' simulation output object resulting from a fitted run.
species.list	A character vector of species names to include in the plots.
scene.name	An optional character string used to label the main title of the plots. Defaults to 'NULL' (resulting in a blank space).

Details

This function opens two separate graphic windows. The first displays biomass fits, iterating through all available data sources for each specified species. The second window displays catch fits. Both plots include an overarching title that dynamically calculates and reports the negative log-likelihood (NLL) of the fit.

Value

Invoked for its side effect of drawing plots to external graphics devices. Returns 'NULL'.

rsim.plot.fitspecies	<i>Plot Full Biomass and Catch Fits for a Single Species Generates a multi-panel figure displaying the simulation fits for a single species across all available biomass observation sources, alongside its catch fitting data.</i>
----------------------	---

Description

Plot Full Biomass and Catch Fits for a Single Species Generates a multi-panel figure displaying the simulation fits for a single species across all available biomass observation sources, alongside its catch fitting data.

Usage

```
rsim.plot.fitspecies(Rsim.scenario, Rsim.output, species)
```

Arguments

Rsim.scenario	An 'Rsim' scenario object containing the fitting data.
Rsim.output	An 'Rsim' simulation output object resulting from a fitted run.
species	A single character string representing the species name.

Details

This function aggregates the biomass and catch visualization into a single row of plots. It dynamically determines the number of biomass data sources for the specified species and configures the graphics layout ('par(mfrow)') to fit all biomass plots plus one additional panel for the catch plot. It calculates log-normal confidence intervals for observations and overlays base reference lines.

Value

Invoked for its side effect of drawing a multi-panel plot to the active graphics device. Returns 'NULL'.

rsim.predprey.table	<i>Extract Predator-Prey Interaction Table</i>
---------------------	--

Description

Generates a data frame of predator-prey linkages and their associated foraging parameters from an 'Rpath' simulation scenario.

Usage

```
rsim.predprey.table(Rsim.scenario, pred = NULL, prey = NULL)
```

Arguments

Rsim.scenario	An 'Rsim' scenario object containing the fitting data.
pred	An optional character string or vector of predator names. If not provided, the function will include all model predators in its output.
prey	An optional character string or vector of prey names. If not provided, the function will include all model prey in its output.

Details

This function outputs a data frame of predator/prey link parameters contained in an Rsim scenario for a given set of predators and prey.

Value

A 'data.frame' containing the columns:

- from, prey name,
- to predator name,
- QQ base consumption rate of the predator/prey link,
- VV vulnerability parameter of the predator/prey link,
- DD handling time parameter of the predator/prey link

rsim.run

*Run a dynamic simulation***Description**

Carries out the numerical integration of the Rsim algorithms.

Usage

```
rsim.run(Rsim.scenario, method = "RK4", years = 1:100, sname = NULL)
```

Arguments

Rsim.scenario	Scenario object that contains all of the rsim rates and forcing functions generated by <code>rsim.scenario()</code> .
method	Numerical integration method. Either 'AB' for Adams-Bashforth or 'RK4' for 4th order Runge-Kutta.
years	A vector of each year of the simulation.
sname	If a single model group name is provided, extra diagnostics on this species will be output.

Value

S3 object class Rsim.output, a list of 16 objects:

- out_Biomass, numeric matrix of monthly (rows) output biomass values by biomass group (columns) for model run
- out_Catch, numeric matrix of monthly (rows) output catch values by biomass group (columns) for model run
- out_Gear_Catch, numeric matrix of monthly (rows) output catch by species/gear/disposition (columns) for model run
- annual_Catch, numeric matrix of annual (rows) output catch values by biomass group (columns) for model run
- annual_Biomass, numeric matrix of annual (rows) output biomass values by biomass group (columns) for model run
- annual_QB, numeric matrix of annual (rows) output consumption:biomass values by biomass group (columns) for model run
- annual_Qlink, numeric matrix of annual (rows) output consumption values by predator-prey pair (columns) for model run
- end_state, list of 9 final value vectors for biomass groups, same components as those produced by `rsim.state()`
- crash_year, integer flag; -1 no crash or year of model crash
- pred, named character vector of predators in each predator prey pair

- prey, named character vector of prey in each predator prey pair
- Gear_Catch_sp, named character vector of species caught in gears for each disposition
- Gear_Catch_gear, named character vector of gear types for each species and disposition
- Gear_Catch_disp, character vector of catch disposition (Landing or Discard) for each species and gear
- start_state, list of 9 initial value vectors for biomass groups, see output of `rsim.state()`
- params, list of 4; 3 dimensioning parameters and 1 biomass group and fleet name vector
- out_species_rates, if spname is provided, this will contain the derivative parts for that species over time

See Also

Other Rsim functions: `extract.node()`, `rsim.params()`, `rsim.plot()`, `rsim.scenario()`, `rsim.step()`, `write.Rsim()`

Examples

```
# Read in Rpath parameter file and generate model object
Rpath <- rpath(AB.params)
# Create a 50 yr Rsim scenario
Rsim.scenario <- rsim.scenario(Rpath, AB.params, years = 1:50)
# Run the Rsim simulation
Rsim.output <- rsim.run(Rsim.scenario, method = "RK4", years = 1:50)
```

rsim.scenario	<i>Specify a dynamic simulation scenario</i>
---------------	--

Description

Uses a static Rpath model and creates a scenario consisting of 5 list objects: params generated by `rsim.params()`, start_state generated by `rsim.state()`, forcing generated by `rsim.forcing()`, fishing generated by `rsim.fishing()`, and 'stanzas' generated by `rsim.stanzas()`.

Usage

```
rsim.scenario(Rpath, Rpath.params, years = 1:100)
```

Arguments

Rpath	R object containing a static Rpath model.
Rpath.params	R object containing the parameters needed to create a Rpath model. This is generated either by the <code>create.rpath.params()</code> or <code>read.rpath.params()</code> functions.
years	A vector of each year of the simulation.

Value

Returns an `Rsim.scenario` object that can be supplied to the `rsim.run()` function. `Rsim.scenario` is a list of 5 objects:

- `params`, list of 39 dynamic parameters and vectors derived from an Rpath model, see output of `rsim.params()`
- `start_state`, list of 9 initial value vectors for biomass and size structured stanza groups, see output of `rsim.state()` and `rsim.stanzas()`
- `forcing`, list of 7 matrices for time series forcing of biological attributes, see output of `rsim.forcing()`
- `fishing`, list of 3 matrices for time series forcing of fishing attributes, see output of `rsim.fishing()`
- `stanzas`, list of 21 parameters and vectors for age structured groups, see output of `rsim.stanzas()`

See Also

Other Rsim functions: `extract.node()`, `rsim.params()`, `rsim.plot()`, `rsim.run()`, `rsim.step()`, `write.Rsim()`

Examples

```
# Read in Rpath parameter file and generate model object
Rpath <- rpath(AB.params)
# Create a 50 yr Rsim scenario
Rsim.scenario <- rsim.scenario(Rpath, AB.params, years = 1:50)
```

rsim.sense

Ecosense function for Rpath

Description

`rsim.sense()` draws a random Rsim parameter set from distributions centered on the supplied Rpath parameter estimates. The width of the distribution for each parameter is defined by the data pedigree of the `Rpath.params` object generated by either `create.rpath.params()` or `read.rpath.params()`. The returned Rsim parameter set is functionally equivalent to a `params` list object created by `rsim.scenario()`.

Usage

```
rsim.sense(Rsim.scenario, Rpath.params, Vvary = c(0, 0), Dvary = c(0, 0))
```

Arguments

Rsim.scenario	The rsim.scenario() object containing the initial rsim parameters used as the center of distributions (base model) during random parameter generation.
Rpath.params	Rpath parameter object (unbalanced model) containing the data pedigree generated by either create.rpath.params() or read.rpath.params().
Vvary	length-2 vector with (lower,upper) bounds of vulnerability parameters generated in log-space - 1, Vvary = log(X-1). Thus, the Walters et al. 1997 vulnerability range of (1,+inf) centered on 2 now becomes (-inf,+inf) centered on 0.
Dvary	length-2 vector with (lower,upper) bounds of handling time generation in log-space - 1. Scaled as Vvary, above (-inf,+inf), centered on 0.

Value

Returns a params list object that can be substituted for the params list in an rsim.scenario object.

References

Walters C, Christensen V, Pauly D (1997) Structuring dynamic models of exploited ecosystems from trophic mass-balance assessments. Rev Fish Biol Fish 7:139-172. doi:10.1023/a:1018479526149

Whitehouse GA, Aydin KY (2020) Assessing the sensitivity of three Alaska marine food webs to perturbations: an example of Ecosim simulations using Rpath. Ecol Model 429:16. doi:10.1016/j.ecolmodel.2020.109074

See Also

vignette("ecosense", package = "Rpath") and Other Rpath functions: [read.rpath.params\(\)](#), [create.rpath.params\(\)](#), [check.rpath.params\(\)](#), [rpath\(\)](#), [rsim.scenario\(\)](#), [rsim.run\(\)](#)

Examples

```
# Read in Rpath parameter file and balance model
EBS_bal <- rpath(Ecosense.EBS)
# Create a 100 yr Rsim scenario
EBS_scene <- rsim.scenario(EBS_bal, Ecosense.EBS, years = 1:100)
# Generate a new params list
new_EBS_params <- rsim.sense(EBS_scene, Ecosense.EBS, Vvary = c(-4.5, 4.5), Dvary = c(-4.5, 4.5))
# assign the generated params to the Rsim.scenario object
EBS_scene$params <- new_EBS_params
```

rsim.stanzas	<i>Initialize list of age structured group (stanza) parameters for dynamic simulation</i>
--------------	---

Description

Creates a list of stanza indices and parameters to be used by rsim.run.

Usage

```
rsim.stanzas(Rpath.params, state, params)
```

Arguments

Rpath.params	R object containing the parameters needed to create a Rpath model. This is generated either by the <code>create.rpath.params()</code> or <code>read.rpath.params()</code> functions.
state	List object of state variables generated by <code>rsim.state</code> .
params	Rsim parameter object generated by <code>rsim.params</code> .

Value

a list of 21 objects:

- `Nsplit`, number of model groups with stanzas
- `Nstanzas`, numeric vector length `Nsplit+1`, leading 0 for indexing followed by the number of age groups for each model group with stanzas
- `EcopathCode`, numeric matrix of Rpath group numbers with `Nsplit+1` species rows and `Nstanzas+1` age group columns, first column and row empty for indexing
- `Age1`, numeric matrix of age in months of first age group with `Nsplit+1` species rows, first column and row empty for indexing, second column earliest age in months (0), third column max age in months
- `Age2`, numeric matrix of age in months of last age group with `Nsplit+1` species rows, first column and row empty for indexing, second column earliest age in months, third column max age in months
- `baseWageS`, numeric matrix with rows of monthly weight relative to weight at infinity (`Winf`) calculated in `rpath.stanzas` with `Nsplit+1` species columns
- `baseNageS`, numeric matrix with rows of monthly numbers surviving, calculated in `rpath.stanzas` with `Nsplit+1` species columns
- `baseQageS`, numeric matrix with rows of monthly consumption, calculated as $WageS^{VBGFd}$ in `rpath.stanzas` with `Nsplit+1` species columns
- `Wmat`, numeric vector length `Nsplit+1`, leading 0 for indexing followed by weight at 50% maturity relative to `Winf` for each model group with stanzas
- `RecPower`, numeric vector length `Nsplit+1`, leading 0 for indexing followed by recruitment power parameter for each model group with stanzas
- `recruits`, numeric vector length `Nsplit+1`, leading 0 for indexing followed by recruits per spawner entered as input to `rpath.stanzas`
- `VBGFd`, numeric vector length `Nsplit+1`, leading 0 for indexing followed by von Bertalanffy d (default = $2/3$) for each model group with stanzas
- `RzeroS`, numeric vector length `Nsplit+1`, leading 0 for indexing followed by recruits per spawner entered as input to `rpath.stanzas`
- `vBM`, numeric vector length `Nsplit+1`, leading 0 for indexing followed by von Bertalanffy growth coefficient for each model group with stanzas

- SplitAlpha, numeric matrix of initial monthly (rows) growth coefficients for each model group with stanzas (Nsplit+1 species columns)
- SpawnX, numeric vector length Nsplit+1, leading 0 for indexing followed by relative strength of spawner-recruit relationship for each model group with stanzas, currently set to 10000, no relationship
- SpawnEnergy, numeric vector length Nsplit+1, leading 0 for indexing followed by relative energy going to recruitment rather than growth for each model group with stanzas, currently set to 1
- baseEggsStanza, numeric vector length Nsplit+1, leading 0 for indexing followed by initial estimate of egg output for each model group with stanzas
- baseSpawnBio, numeric vector length Nsplit+1, leading 0 for indexing followed by initial estimate of spawning biomass (=egg output) for each model group with stanzas
- RscaleSplit, numeric vector length Nsplit+1, leading 0 for indexing followed by adult to juvenile metric scaling parameter for each model group with stanzas, currently set to 1
- baseStanzaPred, numeric vector length NUM_GROUPS+1, base consumption (sum of QageS*NageS) for each model group with stanzas in full model group order; 0s entered for non-stanza groups

Examples

```
# Read in Rpath parameter file and generate model object
Rpath <- rpath(REco.params)
# Create default dynamic parameters from Rpath model
Rsim.params <- rsim.params(Rpath)
# Create state vectors with default values
Rsim.state <- rsim.state(Rsim.params)
# Create stanza object
Rsim.stanzas <- rsim.stanzas(REco.params, Rsim.state, Rsim.params)
```

rsim.state

Initialize starting state matrix for Rsim scenario

Description

Creates a matrix of starting state variables used by Rsim, including initial biomass and numbers by group, as well as feeding time (Ftime). Default initial values: N = 0, Ftime = 1

Usage

```
rsim.state(params)
```

Arguments

params Rsim parameter object generated by rsim.params.

Value

S3 object class `Rsim.state`, a list of 3 objects:

- `Biomass`, numeric vector of initial biomass by biomass group
- `N`, numeric vector of initial numbers by biomass group
- `Ftime`, numeric vector of initial feeding time parameter by biomass group

Examples

```
# Read in Rpath parameter file and generate model object
Rpath <- rpath(AB.params)
# Create dynamic parameters from Rpath model
Rsim.params <- rsim.params(Rpath)
# Create state vectors with default values
Rsim.state <- rsim.state(Rsim.params)
```

`rsim.step`

Function to add a time step to an Rsim run

Description

Runs `rsim.run` and appends the output to a previous `Rsim.output`. Currently only works with the Adams-Bashforth numerical integration method (AB).

Usage

```
rsim.step(Rsim.scenario, Rsim.output, method = "AB", year.end)
```

Arguments

<code>Rsim.scenario</code>	Scenario object that contains all of the <code>rsim</code> rates and forcing functions generated by <code>rsim.scenario()</code> .
<code>Rsim.output</code>	R object containing the output from <code>rsim.run</code> .
<code>method</code>	Numerical integration method. Either 'AB' for Adams-Bashforth or 'RK4' for 4th order Runge-Kutta.
<code>year.end</code>	The final year for the step interval

Value

Returns an `Rsim.output` object.

See Also

Other `Rsim` functions: `extract.node()`, `rsim.params()`, `rsim.plot()`, `rsim.run()`, `rsim.scenario()`, `write.Rsim()`

set.rsim.scene	<i>Set Rsim.scenario parameters</i>
----------------	-------------------------------------

Description

Modifies the various parameters of the `rsim.scenario()` object. Parameters that can be adjusted using this function are: `params`, `start_state`, `forcing`, `fishing`, `stanzas`

Usage

```
set.rsim.scene(
  Rsim.scenario,
  params = NULL,
  start_state = NULL,
  forcing = NULL,
  fishing = NULL,
  stanzas = NULL
)
```

Arguments

<code>Rsim.scenario</code>	Scenario object that contains all of the <code>rsim</code> rates and forcing functions generated by rsim.scenario() .
<code>params</code>	Rsim parameter object generated by <code>rsim.params</code> .
<code>start_state</code>	Rsim starting values object generated by <code>rsim.state()</code>
<code>forcing</code>	Rsim forcing matrix object generated by <code>rsim.forcing()</code>
<code>fishing</code>	Rsim fishing matrix object generated by <code>rsim.fishing()</code>
<code>stanzas</code>	Rsim stanza parameters object generated by <code>rsim.stanzas()</code>

Value

Returns an `Rsim.scenario` object with the new parameter.

See Also

Other Adjust functions: [adjust.fishing\(\)](#), [adjust.forcing\(\)](#), [adjust.scenario\(\)](#)

Examples

```
# Read in Rpath parameter file and generate balanced model
Rpath <- rpath(AB.params)
# Create a 50 yr Rsim scenario
Rsim.scenario <- rsim.scenario(Rpath, AB.params, years = 1:50)
stanzas <- rsim.stanzas(AB.params)
Rsim.scenario.new <- set.rsim.scene(Rsim.scenario, stanzas=stanzas)
```

stanzaplot

Plot routine for Ecopath multistanzas

Description

Plots the biomass composition of multistanza groups from an Rpath.stanzas object.

Usage

```
stanzaplot(
  Rpath.params,
  StanzaGroup,
  line.cols = c("black", "green", "blue", "red")
)
```

Arguments

Rpath.params	R object containing the parameters needed to create a Rpath model. This is generated either by the create.rpath.params() or read.rpath.params() functions.
StanzaGroup	The Stanza group's name to be plotted. Valid values are found in the 'stanzas' field of the Rpath parameter file.
line.cols	A vector of four colors used to represent the population biomass, relative number, individual weights, and stanza separation lines.

Value

Creates a figure showing the break down of biomass and number per stanza.

See Also

Other Rpath functions: [MTI\(\)](#), [check.rpath.params\(\)](#), [create.rpath.params\(\)](#), [frate.table\(\)](#), [read.rpath.params\(\)](#), [rpath\(\)](#), [rpath.consumers\(\)](#), [rpath.detrital\(\)](#), [rpath.gears\(\)](#), [rpath.groups\(\)](#), [rpath.living\(\)](#), [rpath.mixotrophs\(\)](#), [rpath.producers\(\)](#), [rpath.stanzas\(\)](#), [webplot\(\)](#), [write.Rpath\(\)](#), [write.rpath.params\(\)](#)

Examples

```
# Choose group with multiple stanzas
params <- REco.params
stanzaplot(params, StanzaGroup = "Roundfish1")
```

webplot

*Plot routine for Ecopath food web***Description**

Plots the food web associated with an Rpath object.

Usage

```
webplot(
  Rpath.obj,
  eco.name = attr(Rpath.obj, "eco.name"),
  line.col = "grey",
  highlight = NULL,
  highlight.col = c("black", "red", "orange"),
  labels = FALSE,
  label.pos = NULL,
  label.num = FALSE,
  label.cex = 1,
  fleets = FALSE,
  type.col = "black",
  box.order = NULL
)
```

Arguments

Rpath.obj	Rpath model created by rpath().
eco.name	Optional name of the ecosystem. Default is the 'eco.name' attribute from the rpath object created from running rpath().
line.col	The color of the lines between nodes of the food web.
highlight	Group number or name to highlight the connections of that group. Valid values are found in the 'Group' field of the object created from running rpath().
highlight.col	Color of the connections to the highlighted group, vector of length 3. Defaults to black for prey, red for predator, orange for fleet.
labels	Logical whether or not to display group names. If TRUE and label.pos = NULL, no points will be plotted, just label names.
label.pos	A position specifier for the labels. Values of 1, 2, 3, 4 indicate positions below, to the left of, above, and to the right of the points, respectively. A null value will cause the labels to be plotted without the points, assuming that labels = TRUE.
label.num	Logical value indication whether group numbers should be used for labels instead of names.
label.cex	Numeric value of the relative size of the labels within the plot.
fleets	Logical value indicating whether or not to include fishing fleets in the food web plot.

type.col	The color of the points corresponding to the types of the group. Must be of length 1 or 4. Color order will be consumers, primary producers, detrital, and fleet groups.
box.order	Vector of box numbers to change the default plot order. Must include all box numbers

Value

Returns a plot visualization of the food web.

See Also

Other Rpath functions: [MTI\(\)](#), [check.rpath.params\(\)](#), [create.rpath.params\(\)](#), [frate.table\(\)](#), [read.rpath.params\(\)](#), [rpath\(\)](#), [rpath.consumers\(\)](#), [rpath.detrital\(\)](#), [rpath.gears\(\)](#), [rpath.groups\(\)](#), [rpath.living\(\)](#), [rpath.mixotrophs\(\)](#), [rpath.producers\(\)](#), [rpath.stanzas\(\)](#), [stanzaplot\(\)](#), [write.Rpath\(\)](#), [write.rpath.params\(\)](#)

Examples

```
# Read in Rpath parameter file, generate and name model object
Rpath.obj <- rpath(AB.params, eco.name = "Anchovy Bay")
# Plot food web diagram with all groups labeled, including fleets
webplot(Rpath.obj, labels = TRUE, fleets = TRUE)
# Plot food web diagram without labels, highlighting connections of cod group
webplot(Rpath.obj, highlight = "cod", fleets = TRUE)
```

write.Rpath	<i>Write function for Rpath food web object</i>
-------------	---

Description

Outputs basic parameters or mortalities from a `rpath()` object to a .csv or .RData file.

Usage

```
write.Rpath(x, file = NA, morts = F)
```

Arguments

x	Rpath model object generated by <code>rpath()</code> .
file	file name for resultant file. Need to specify ".csv" or ".RData".
morts	Logical value whether to output basic parameters or mortalities.

Value

Writes a list object or file with the basic parameters or mortalities from an Rpath object.

If `morts = F`, output reflects the basic parameters of the mass-balanced model, namely:

Group	Vector of group names
type	Vector of group types: 0 = consumer, 1 = producer, 2 = detritus, 3 = fishery
TL	Vector of trophic level for each group, calculated based on mass balance
Biomass	Vector of biomass values for each modeled group
PB	Vector of production:biomass ratios for each modeled group
QB	Vector of consumption:biomass ratios for each modeled group
EE	Vector of ecotrophic efficiency values for each modeled group
GE	Vector of gross efficiency values (production:consumption ratios) for each modeled group
Removals	Vector of total fishery removals for each modeled group

If `morts = T`, output reflects calculated mortality values for each group. The number of columns will vary based on how many predators are in the model.

Group	Vector of group names
M0	Vector of natural mortality for each group. Natural mortality = $PB * (1 - EE)$
Fmort	Vector of fishing mortality for each group
M2.XX	Vector of predation mortality, where XX is the name of a predator group

See Also

Other Rpath functions: [MTI\(\)](#), [check.rpath.params\(\)](#), [create.rpath.params\(\)](#), [frate.table\(\)](#), [read.rpath.params\(\)](#), [rpath\(\)](#), [rpath.consumers\(\)](#), [rpath.detrital\(\)](#), [rpath.gears\(\)](#), [rpath.groups\(\)](#), [rpath.living\(\)](#), [rpath.mixotrophs\(\)](#), [rpath.producers\(\)](#), [rpath.stanzas\(\)](#), [stanzaplot\(\)](#), [webplot\(\)](#), [write.rpath.params\(\)](#)

Examples

```
## Not run:
# Read in Rpath parameter file, generate model object
x <- rpath(AB.params)
# Saves basic model parameters for all groups to the root directory of users project
write.Rpath(x, file = "output.csv", morts = F)
# Calculates natural mortality (M0), fishing mortality by fleet, and predation mortality by predator (M2) on each group
# Writes output to an Rdata file in the root directory of users project
write.Rpath(x, file = here::here("output.RData"), morts = T)

## End(Not run)
```

write.rpath.params	<i>Write Rpath parameters to .csv files</i>
--------------------	---

Description

Creates a series of .csv files from an Rpath.params object.

Usage

```
write.rpath.params(Rpath.params, eco.name, path = "")
```

Arguments

Rpath.params	R object containing the Rpath parameters. Most likely this was created using create.rpath.params() or read.rpath.params() .
eco.name	ecosystem name that will be included in all the file names.
path	location for the output files.

Value

Outputs a series of .csv files named by the provided eco.name and the parameters they represent. For example the model parameters will be named "eco.name_model.csv".

See Also

Other Rpath functions: [MTI\(\)](#), [check.rpath.params\(\)](#), [create.rpath.params\(\)](#), [frate.table\(\)](#), [read.rpath.params\(\)](#), [rpath\(\)](#), [rpath.consumers\(\)](#), [rpath.detrital\(\)](#), [rpath.gears\(\)](#), [rpath.groups\(\)](#), [rpath.living\(\)](#), [rpath.mixotrophs\(\)](#), [rpath.producers\(\)](#), [rpath.stanzas\(\)](#), [stanzaplot\(\)](#), [webplot\(\)](#), [write.Rpath\(\)](#)

write.Rsim	<i>Write function for Rsim run</i>
------------	------------------------------------

Description

Outputs starting biomass, ending biomass, and catch to a .csv or .Rdata file.

Usage

```
write.Rsim(Rsim.output, file = NA)
```

Arguments

Rsim.output	object created by the <code>rsim.run()</code> function.
file	file name for resultant file. Need to specify ".csv" or ".RData".

Value

Writes a list object or file with the start and end biomass and catch per group from an Rpath.sim object.

Group	Vector of group names
StartBio	Vector of Biomass values at simulation start
EndBio	Vector of Biomass values at simulation end
BioES	Vector of ratio end:start Biomass values
StartCatch	Vector of Catch values at simulation start
EndCatch	Vector of Catch values at simulation end
CatchES	Vector of ratio end:start Catch values

See Also

Other Rsim functions: [extract.node\(\)](#), [rsim.params\(\)](#), [rsim.plot\(\)](#), [rsim.run\(\)](#), [rsim.scenario\(\)](#), [rsim.step\(\)](#)

Examples

```
# Read in Rpath parameter file and generate model object
Rpath <- rpath(AB.params)
# Create a 50 yr Rsim scenario
Rsim.scenario <- rsim.scenario(Rpath, AB.params, years = 1:50)
# Run the Rsim simulation
Rsim.output <- rsim.run(Rsim.scenario, method = "RK4", years = 1:50)
# Save summary output to a .csv file
write.Rsim(Rsim.output, file = "output.csv")
```

Index

- * **Adjust functions**
 - adjust.fishing, 4
 - adjust.forcing, 5
 - adjust.scenario, 6
 - set.rsim.scene, 61
- * **Get functions**
 - get.rsim.fishing, 16
 - get.rsim.forcing, 16
 - get.rsim.params, 17
 - get.rsim.stanzas, 18
 - get.rsim.start_state, 19
- * **It has been tested to give the same results as Whitehouse and Aydin (2020)**
 - rsim.sense, 56
- * **Rpath functions**
 - check.rpath.params, 7
 - create.rpath.params, 9
 - frate.table, 14
 - MTI, 22
 - read.rpath.params, 24
 - rpath, 26
 - rpath.consumers, 26
 - rpath.detrital, 27
 - rpath.gears, 28
 - rpath.groups, 29
 - rpath.living, 30
 - rpath.mixotrophs, 30
 - rpath.producers, 31
 - rpath.stanzas, 32
 - rsim.sense, 56
 - stanzaplot, 62
 - webplot, 63
 - write.Rpath, 64
 - write.rpath.params, 66
- * **Rsim functions**
 - extract.node, 12
 - rsim.params, 47
 - rsim.plot, 50
 - rsim.run, 54
 - rsim.scenario, 55
 - rsim.step, 60
 - write.Rsim, 66
- * **datasets**
 - AB.params, 3
 - Ecosense.EBS, 9
 - Ecosense.ECS, 10
 - Ecosense.GOA, 11
 - REco.params, 25
- * **previous version (previous version is rsim.sense.path)**
 - rsim.sense, 56
- * **random seed - though some minor bug corrections have been found in the**
 - rsim.sense, 56
- * **when supplied with a straight-from-ecopath rsim scenario. and the same**
 - rsim.sense, 56
- * **xml**
 - create.rpath.from.eiixml, 8
 - import.eiixml, 21
- *
 - rsim.sense, 56
- AB.params, 3
- adjust.fishing, 4
- adjust.fishing(), 6, 7, 61
- adjust.forcing, 5
- adjust.forcing(), 5, 7, 61
- adjust.scenario, 6
- adjust.scenario(), 5, 6, 61
- check.rpath.params, 7, 57
- check.rpath.params(), 9, 15, 22, 25–32, 62, 64–66
- create.rpath.from.eiixml, 8
- create.rpath.from.eiixml(), 21
- create.rpath.params, 7, 9, 22, 26, 32, 55, 57, 58, 62, 66

- create.rpath.params(), 7, 15, 22, 25–32, 62, 64–66
- Ecosense.EBS, 9
- Ecosense.ECS, 10
- Ecosense.GOA, 11
- extract.node, 12
- extract.node(), 50, 51, 55, 56, 60, 67
- fitcatch.to.forcecatch, 13
- frate.table, 14
- frate.table(), 7, 9, 22, 25–32, 62, 64–66
- get.pp.link, 15
- get.rsim.fishing, 16
- get.rsim.fishing(), 17–19
- get.rsim.forcing, 16
- get.rsim.forcing(), 16–19
- get.rsim.params, 17
- get.rsim.params(), 16–19
- get.rsim.stanzas, 18
- get.rsim.stanzas(), 16, 17, 19
- get.rsim.start_state, 19
- get.rsim.start_state(), 16–18
- ggwebplot, 19
- import.eiixml, 21
- import.eiixml(), 8
- MTI, 22
- MTI(), 7, 9, 15, 25–32, 62, 64–66
- read.fitting.biomass, 23
- read.fitting.catch, 23
- read.rpath.params, 7, 22, 24, 26, 32, 55, 57, 58, 62, 66
- read.rpath.params(), 7, 9, 15, 22, 26–32, 62, 64–66
- REco.params, 25
- rpath, 26, 57
- rpath(), 7, 9, 15, 22, 25, 27–32, 62, 64–66
- rpath.consumers, 26
- rpath.consumers(), 7, 9, 15, 22, 25–32, 62, 64–66
- rpath.detrital, 27
- rpath.detrital(), 7, 9, 15, 22, 25–32, 62, 64–66
- rpath.gears, 28
- rpath.gears(), 7, 9, 15, 22, 25–27, 29–32, 62, 64–66
- rpath.groups, 29
- rpath.groups(), 7, 9, 15, 22, 25–28, 30–32, 62, 64–66
- rpath.living, 30
- rpath.living(), 7, 9, 15, 22, 25–29, 31, 32, 62, 64–66
- rpath.mixotrophs, 30
- rpath.mixotrophs(), 7, 9, 15, 22, 25–32, 62, 64–66
- rpath.producers, 31
- rpath.producers(), 7, 9, 15, 22, 25–32, 62, 64–66
- rpath.stanzas, 32
- rpath.stanzas(), 7, 9, 15, 22, 25–31, 62, 64–66
- rsim.deriv, 33
- rsim.diet, 34
- rsim.fishing, 35, 55, 56
- rsim.fit.apply, 36
- rsim.fit.list.bio.bysurvey, 37
- rsim.fit.list.bio.series, 37
- rsim.fit.list.catch.series, 38
- rsim.fit.obj, 39
- rsim.fit.obj.species, 39
- rsim.fit.remove.bio.timeseries, 40
- rsim.fit.run, 41
- rsim.fit.set.bio.wt, 42
- rsim.fit.set.catch.wt, 43
- rsim.fit.set.q, 43
- rsim.fit.table, 44
- rsim.fit.update, 45
- rsim.forcing, 46, 55, 56
- rsim.mort, 47
- rsim.params, 47, 55, 56
- rsim.params(), 13, 51, 55, 56, 60, 67
- rsim.plot, 50
- rsim.plot(), 13, 50, 55, 56, 60, 67
- rsim.plot.fit, 51
- rsim.plot.fitspecies, 52
- rsim.predprey.table, 53
- rsim.run, 34, 47, 54, 56, 57
- rsim.run(), 13, 50, 51, 56, 60, 67
- rsim.scenario, 4–6, 14, 16–19, 33, 54, 55, 57, 60, 61
- rsim.scenario(), 13, 50, 51, 55, 60, 67
- rsim.sense, 56
- rsim.stanzas, 55, 56, 57
- rsim.state, 54–56, 59

`rsim.step`, 60
`rsim.step()`, 13, 50, 51, 55, 56, 67

`set.rsim.scene`, 61
`set.rsim.scene()`, 5–7
`stanzaplot`, 62
`stanzaplot()`, 7, 9, 15, 22, 25–32, 64–66

`webplot`, 63
`webplot()`, 7, 9, 15, 22, 25–32, 62, 65, 66
`write.Rpath`, 64
`write.Rpath()`, 7, 9, 15, 22, 25–32, 62, 64, 66
`write.rpath.params`, 66
`write.rpath.params()`, 7, 9, 15, 22, 25–32,
62, 64, 65
`write.Rsim`, 66
`write.Rsim()`, 13, 50, 51, 55, 56, 60