Package: FLBEIA (via r-universe)

August 16, 2024

Title Bio-Economic Impact Assessment of Management Strategies using FLR

Version 1.16.1.19

Date 2024-07-16

Description A simulation toolbox that describes a fishery system under a Management Strategy Estrategy approach. The objective of the model is to facilitate the Bio-Economic evaluation of Management strategies. It is multistock, multifleet and seasonal. The simulation is divided in 2 main blocks, the Operating Model (OM) and the Management Procedure (MP). In turn, each of these two blocks is divided in 3 components: the biological, the fleets and the covariables on the one hand, and the observation, the assessment and the advice on the other.

LinkingTo Rcpp

Depends methods, R(>= 3.5.0), ggplot2, FLCore (>= 2.6.5), FLFleet

Imports stats, utils, FLXSA, FLa4a, Matrix, FLash, mvtnorm, mlogit, nloptr, triangle, arrayhelpers, dplyr, plyr, purrr, tidyr, tidyverse, tibble, Rcpp

Suggests R.rsp, spict, XLConnect

Additional_repositories http://flr-project.org/R

License GPL (>= 2)

Language en-GB

URL http://flr-project.org/FLBEIA

BugReports https://github.com/flr/FLBEIA/issues

Collate ``genericMethods.R'' ``Class_0a_FLlst-beia.R'' ``Class_0b_FLAccesors_beia.R'' ``Class_1_FLBDsim.R'' ``Class_2_FLSRsim.R'' ``Class_3_FLCatchExt.R'' ``Class_4_FLMetierExt.R'' ``Class_5_FLFleetExt.R'' ``Auxiliary_Functions_biolfleets2flstock.R'' ``Auxiliary_Functions_Calculations_exem.R'' ``Auxiliary_Functions_Checks.R''

``Auxiliary_Functions_Data_Manipulation.R''
``Auxiliary_Functions_Effort_Dynamics.R''
``Auxiliary_Functions_Observation_Model.R''
``Auxiliary_Functions_Population_Dynamics.R''
``Auxiliary_Functions_remove_units.R''
``Auxiliary_Functions_xtabs2.R'' ``setPlusGroupFLBiol.R''
``stf_correctSel.R'' ``Conditioning_1_CobbDouglasFit.R''
``Conditioning_2_SelAtAge.R'' ``Conditioning_calculate.CDparam.R''
``Conditioning_create.advice.data.R''
``Conditioning_create.BDs.data.R''
``Conditioning_create.biol.arrays.R''
``Conditioning_create.biols.data.R''
``Conditioning_create.ecoData.arrays.R''
``Conditioning_create.fleets.arrays.R''
``Conditioning_create.fleets.data.R''
``Conditioning_create.indices.data.R''
``Conditioning_create.list.stks.flq.R''
``Conditioning_create.list.stks.flqa.R''
``Conditioning_create.SRs.data.R''
``Conditioning_CtrlCreators_advice.r''
``Conditioning_CtrlCreators_assessment.r''
``Conditioning_CtrlCreators_biols.R''
``Conditioning_CtrlCreators_covars.r''
``Conditioning_CtrlCreators_fleets.r''
``Conditioning_CtrlCreators_Observation.r''
``Conditioning_InitialPop.R'' ``data.R'' ``ecoSum_DAMARA_function.R''
``Functions_Checks.R'' ``Functions_FLFleet.R''
``Functions_FLFleets.R'' ``MAIN_Function.R''
``Method_propagate_FLFleet.R'' ``MP_1_Observation.R''
``MP_1a_Observation_Model.R''
``MP_1b_Observation_Model_Functions.R'' ``MP_2_Assessment.R''
``MP_2_Assessment_sca2flbeia.R'' ``MP_2_Assessment_spict2flbeia.R''
``MP_3_Advice.R'' ``MP_3a_fwdBD.R'' ``MP_3a_fwdBDControl.R''
``MP_3b_HCRs.R'' ``MP_3c_HCR_BoBane.R'' ``MP_3c_HCR_BoBane_JD.R''
``MP_3c_HCR_F2CatchHCR.R'' ``MP_3c_HCR_Froese.R''
``MP_3c_HCR_ICES_MSY.R'' ``MP_3c_HCR_neaMAC_ltmp.R''
``MP_3c_HCRE_ane.R'' ``MP_3d_HCR_AnnexIV.R'' ``MP_3d_HCR_GHL.R''
``MP_3d_HCR_ICES_Cat3.R'' ``MP_3d_HCR_ICES_Cat3_bsafe_hrcap.R''
``MP_3e_HCR_SWWMAP.R'' ``MP_3f_CFP_MSY_HCR.R''
``MP_3g_MultiStockHCR.R'' ``MP_3h_ModelFreeHCR.R'' ``OM_1_biol.om.R''
``OM_1a_Population_Growth_Functions.R''
``OM_1a1_Stock_Recruitment_functions.R''
``OM_1a1_Stock_Recruitment_functions_segregmix.R''
``OM_1a2_DensityDependent_weight_functions.R'' ``OM_2_fleet.om.R''
``OM_2a_Effort_Dynamics.R'' ``OM_2a_Effort_Dynamics_Auxiliary.R''
``OM_2a_Effort_Dynamics_LO_functions_exem.R''
``OM_2a_Effort_Dynamics_LO_MaxProf.R''
``OM_2a_Effort_Dynamics_LO_QuotaSwap_exem.R''

``OM_2a_Effort_Dynamics_MaxProf.R"
``OM_2a_Effort_Dynamics_SMFB.R"
``OM_2a_Effort_Dynamics_SMFB_VarEffortShare.R"
``OM_2a1_Catch_Production_Functions.R"
``OM_2a1_catchability2zero.R" ``OM_2b_EffDyn_TAC_OverS_Func.R"
``OM_2b_Update_catch.R" ``OM_2c_Price_Dynamics.R"
``OM_2d_Fleet_Capital_Dynamics.R" ``OM_2e_CapitalDyn_SCD.R"
``OM_3_covars.om.R" ``PlotEco.R" ``PlotFLBiols.R" ``PlotFLFleets.R"
``PlotfltStkSum.R" ``Results_JoinIterations.R"
``Results_Summary.R" ``Results_SummaryPlots.R" ``brps_seasonal.R"
``RcppExports.R"

VignetteBuilder R.rsp

RoxygenNote 7.2.2

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

RemoteUrl https://github.com/flr/FLBEIA

RemoteRef HEAD

RemoteSha 3936481dc2b2ad1150cba2aed2a2dc7e8afe5863

Contents

`annualTAC` *Harvest Control Rules (HCRs)*

Description

There are several HCRs available in FLBEIA which are used within the main function FLBEIA to generate the management advice in each step. But they can also be used independently.

Usage

```
annualTAC(stocks, advice, advice.ctrl, year, stknm, ...)

F2CatchHCR(stocks, advice, advice.ctrl, year, stknm, ...)

FroeseHCR(stocks, advice, advice.ctrl, year, stknm, ...)

IcesHCR(stocks, advice, advice.ctrl, year, stknm, ...)

neaMAC_ltmp(stocks, advice, advice.ctrl, year, stknm, ...)

aneHCRE(stocks, advice, advice.ctrl, year, stknm, ...)
```

```
annexIVHCR(indices, advice, advice.ctrl, year, stknm, ...)

ghlHCR(indices, advice, advice.ctrl, year, stknm, ...)

IcesCat3HCR(indices, advice, advice.ctrl, year, stknm, ...)

IcesCat3HCR_bsafe_hrcap(indices, advice, advice.ctrl, year, stknm, ...)

MAPHCR(stocks, advice, advice.ctrl, year, stknm, ...)

CFPMSYHCR(stocks, advice, advice.ctrl, year, stknm, ...)

MultiStockHCR(stocks, indices, advice, advice.ctrl, year, stknm, ...)

pidHCR(indices, advice, advice.ctrl, year, stknm, ...)

pidHCRItarg(indices, advice, advice.ctrl, year, stknm, ...)

little2011HCR(indices, advice, advice.ctrl, year, stknm, ...)
```

Arguments

stocks	And FLStocks object.
advice	A list with two FLQuant elements, TAC and quota.share. TAC is an FLQuant with quant dimension equal to the number of stocks in biols object, the names used in in the quant dimension must be equal to those used in biols. quota.share is a list with one element per stock in biols object indicating the quota share per stock and fleet. The quant dimension of the elements must be equal to the number of fleets and the names used must be equal to those in fleets objects.
advice.ctrl	A list with the settings to control the advice model for each stock (the HCR for each stock, the reference points used in the HCR, additional parameters, ...)
year	The position of the assessment year in the stocks and advice objects.
stknm	The name of the stock for which advice is being generated.
...	Any extra arguments needed for specific HCRs.
indices	A list of FLIndices. Each element must correspond with one of the stocks in biols object.

Details

There are two types of HCRs model-free HCRs and model-Based HCRs. Model-free HCRs use abundance indices to generate the advice and hence it use FLIndices object as input data. Model-based HCRs use estimates of stock abundance and stock exploitation level to generate the advice.

- Model-Free HCRs: annexIVHCR, ghlHCR, little2011HCR, pidHCR, pidHCRItarg, IcesCat3HCR and IcesCat3HCR_bsafe_hrcap.
- Model-Based HCRs: aneHCRE, annualTAC, CFPMSYHCR, F2CatchHCR, FroeseHCR, IcesHCR, MAPHCR, neaMAC_ltmp.

Rules' description:

- aneHCRE: The HCR used in the bay of biscay anchovy long term management plan.
- aneHCRs: The HCRs (escapement biomass) tested for the bay of biscay anchovy with different calendars. For details see Sanchez et al. 2019. MEPS 617-618: 245-263.
- annexIVHCR: The HCR used by EC and ICES to generate the TAC advice for data poor stocks.
- annualTAC: A HCR that generates annual TAC advice. The HCR provides the whole flexibility of fwd.
- CFPMSYHCR: HCR adapting the MAP HCR to allow flexibility in the year Fmsy is achieved. The user can specify the year in which you aim to reach Fmsy, with a linear transition between Fsq to Fmsy in the intervening years
- F2CatchHCR: This HCR transforms the fishing mortality advice given as input data to catch advice without any other restriction.
- FroeseHCR: The HCR defined in the paper by Froese, Branch et al. in Fish and Fisheries 2011.
- ghlHCR: The model-free HCR used in the management of greenland-halibut
- IcesHCR: The HCR used by ICES to generate TAC advice in the MSY framework.
- IcesCat3HCR: HCR that implements the ICES HCR for Category 3 DLS stocks (see ICES CM 2012/ACOM 68: Category 3 - Method 3.2)
- IcesCat3HCR_bsafe_hrcap: Alternative approach for IcesCat3HCR with a biomass safeguard and harvest rate caps (from ICES WKDLSSLS2019)
- little2011HCR: The HCR defined in the paper by Little et al. in ICES Journal of Marine Science 2011.
- MAPHRC: The HCR proposed by the EC in the evaluation on multi-annual management plans in 2015.
- MultiStockHRC: A HCR that produces TAC advice for several stocks simultaneously. It uses a fishing mortality target and an upper bound to conciliate the TAC advices. In the case of stocks without exploitation rate estimates it uses the catch.
- neaMAC_ltmp: The HCR used in the north-east atlantic mackerel long term management plan. It is a particular case of the IcesHCR.
- pidHCR, pidHCRItarg: The HCRs defined in the paper by Pomaerede et al. in Aquatic Living Resources 2010.

The HCRs are documented in detail in the manual of the library.

Value

The advice input object updated with the management advice (TAC) generated by the HCR.

Examples

```
## Not run:
library(FLBEIA)
library(FLAssess)          # required to use the IcesHCR. Not available for win64
```

```
library(FLash)             # required to use the IcesHCR. Not available for win64
library(ggplot2)

# Load the data to run FLBEIA in a one stock one fleet example using the HCR used by ICES
# in the MSY framework.
 data(one)

oneAdv$TAC[,ac(2009:2025)] <- NA # Put NA-s in the projection years to check how the
                                 # function fills the advice object.

res <- IcesHCR(oneSt, oneAdv, oneAdvC, 19, 'stk1')
# The value printed in the screen is the fishing mortality used in the advice.

res$TAC[,'2009']    # The resulting management advice.

## End(Not run)
```

`biolfleets2flstock` *Function to generate an FLStock from an FLBiol and FLFleet*

Description

Function to generate an FLStock from an FLBiol and FLFleet

Usage

```
biolfleets2flstock(biol, fleets)
```

Arguments

`biol`	An FLBiolobject
`fleets`	An FLFleetExt object

Value

An FLStock object with abundance and biological information from biol argument and exploitation levels from fleets argument

Examples

```
## Not run:

data(res_flbeia)

biol   <- oneRes$biols$stk1
fleets <- oneRes$fleets

stk <- biolfleets2flstock( biol=biol, fleets=fleets)


## End(Not run)
```

bioSum *Summary of the FLBEIA output*

Description

Summarize the results of the simulation in data frames.

Usage

```
bioSum(
  obj,
  stknms = "all",
  years = dimnames(obj$biols[[1]]@n)$year,
  long = FALSE,
  scenario = "bc",
  byyear = TRUE,
  ssb_season = NULL,
  brp = NULL
)

bioSumQ(obj, prob = c(0.95, 0.5, 0.05))

fltSum(
  obj,
  flnms = "all",
  years = dimnames(obj$biols[[1]]@n)$year,
  byyear = TRUE,
  long = FALSE,
  InterestRate = 0.03,
  scenario = "bc"
)

fltSumQ(obj, prob = c(0.95, 0.5, 0.05))

fltStkSum(
  obj,
  flnms = names(obj$fleets),
  stknms = catchNames(obj$fleets),
  years = dimnames(obj$biols[[1]]@n)[[2]],
  byyear = TRUE,
  long = FALSE,
  scenario = "bc",
  verbose = TRUE
)

fltStkSumQ(obj, prob = c(0.95, 0.5, 0.05))
```

```
mtStkSum(
  obj,
  flnms = names(obj$fleets),
  stknms = catchNames(obj$fleets),
  years = dimnames(obj$biols[[1]]@n)[[2]],
  byyear = TRUE,
  long = FALSE,
  scenario = "bc"
)

mtStkSumQ(obj, prob = c(0.95, 0.5, 0.05))

mtSum(
  obj,
  flnms = names(obj$fleets),
  years = dimnames(obj$biols[[1]]@n)[[2]],
  byyear = TRUE,
  long = FALSE,
  scenario = "bc"
)

mtSumQ(obj, prob = c(0.95, 0.5, 0.05))

advSum(
  obj,
  stknms = "all",
  years = dimnames(obj$biols[[1]]@n)$year,
  long = FALSE,
  scenario = "bc"
)

advSumQ(obj, prob = c(0.95, 0.5, 0.05))

riskSum(
  obj,
  stknms = names(obj$biols),
  Bpa,
  Blim,
  Prflim,
  flnms = names(obj$fleets),
  years = dimnames(obj$biols[[1]]@n)[[2]],
  scenario = "bc"
)

npv(
  obj,
  discF = 0.05,
  y0,
```

```
  flnms = names(obj$fleets),
  years = dimnames(obj$biols[[1]]@n)[[2]],
  scenario = "bc"
)

npvQ(obj, prob = c(0.05, 0.5, 0.95))

vesselSum(
  obj,
  flnms = "all",
  years = dimnames(obj$biols[[1]]@n)$year,
  byyear = TRUE,
  long = FALSE,
  scenario = "bc"
)

vesselSumQ(obj, prob = c(0.95, 0.5, 0.05))

vesselStkSum(
  obj,
  flnms = names(obj$fleets),
  stknms = catchNames(obj$fleets),
  years = dimnames(obj$biols[[1]]@n)[[2]],
  byyear = TRUE,
  long = FALSE,
  scenario = "bc"
)

vesselStkSumQ(obj, prob = c(0.95, 0.5, 0.05))
```

Arguments

obj	The output of the FLBEIA function.
stknms	Names of the stock for which the indicators will be calculated.
years	the names of the years for which the indicators will be calculated.
long	logical. The data frame should be constructed using long or wide format? Default FALSE.
scenario	a character string with the name of the scenario corresponding with obj. Default bc.
byyear	logical. The indicators should be provided at season or year level? Default TRUE.
ssb_season	If byyear = TRUE, the season in which ssb will be taken.
brp	a data frame with columns stock, iter and one colum per reference point with the value of the biological reference points per stock and iteration. The used reference points are Bpa, Blim, Btarget, Fpa, Flim and Ftarget.
prob	a numeric vector with the probabilities used to calculate the quantiles.

flnms	Names of the fleet for which the indicators will be calculated.
InterestRate	Capital oportunity cost rate.
verbose	logical. If TRUE, prints the function steps.
Bpa	named numeric vector with one element per stock in stknms. The precautionary approach stock spawning biomass used in riskSum function to calculate biological risk yearly.
Blim	named numeric vector with one element per stock in stknms. The limit stock spawning biomass used in riskSum function to calculate biological risk yearly.
discF	Discount rate.
y0	character. Reference year.
ProfRef	named numeric vector with one element per fleet in flnms. The reference profit level used in riskSum function to calculate economic risk yearly.

Details

- advSum, advSumQ: Data frame with the indicators related with the management advice (TAC). The indicators are: "catch", "discards", "discRat", "landings", "quotaUpt" and "tac".
- bioSum, bioSumQ: Data frame with the biological indicators. The indicators are: "biomass", "catch", "catch.iyv", "discards", "disc.iyv", "f", "landings", "land.iyv", "rec" and "ssb".
- fltSum, fltSumQ: Data frame with the indicators at fleet level. The indicators are: "capacity", "catch", "costs", "discards", "discRat", "effort", "fcosts", "gva", "grossValue", "landings", "fep", "nVessels", "price", "grossSurplus", "quotaUpt", "salaries", "vcosts" and "profitability".
- fltStkSum, fltStkSumQ: Data frame with the indicators at fleet and stock level. The indicators are: "landings", "discards", "catch", "price", "quotaUpt", "tacshare", "discRat" and "quota".
- npv: A data frame with the net present value per fleet over the selected range of years.
- mtSum, mtSumQ: Data frame with the indicators at metier level. The indicators are: "effshare", "effort", "grossValue" and "vcost".
- mtStkSum, mtStkSumQ: Data frame with the indicators at fleet and metier level. The indicators are: "catch", "discards", "discRat", "landings" and "price".
- riskSum: A data frame with the risk indicators. The indicators are: "pBlim", "pBpa" and "pPrflim".
- vesselSum, vesselSumQ: Data frame with the indicators at vessel level. The indicators are: "catch", "costs", "discards", "discRat", "effort", "fcosts", "gva", "grossValue", "landings", "fep", "price", "grossSurplus", "quotaUpt", "salaries", "vcosts" and "profitability".
- vesselStkSum, vesselStkSumQ: Data frame with the indicators at vessel and stock level. The indicators are: "landings", "discards", "catch", "price", "quotaUpt", "tacshare", "discRat" and "quota".
- summary_flbeia: An array with four dimensions: stock, year, iteration, indicator. The indicators are: recruitment, ssb, f, biomass, catch, landings and discards.
- ecoSum_damara: ecoSum built in the framework of Damara project.

The data frames

Value

A data frame with columns for scenario, year, stock, iter, indicator, value,...

The data frames can be of wide or long format. In long format all the indicators are in the same column. There is one column, indicator, for the name of the indicator and a second one value for the numeric value of the indicator. In the wide format each of the indicators correspond with one column in the data frame. The long format it is recommendable to work with ggplot2 functions for example while the wide format it is more efficient for memory allocation and speed of computations.

The quantile version of the summaries, fooQ, returns the quantiles of the indicators. In the long format as many columns as elements in prob are created. The name of the columns are the elements in prob preceded by a q. In the wide format for each of the indicators as many columns as elements in prob are created. The names of the colums are the elements in prob preceded by q_name_of_the_indicator.

Examples

```
## Not run:

library(FLBEIA)

# Apply the summary functions to the examples runs in FLBEIA help page.
# Test the different arguments in summary function.

data(res_flbeia)


#------------------------------------------------
# Example One: One stock, one fleet, one iter.
#------------------------------------------------

# Wide format (default)

oneRes_bio    <- bioSum(oneRes)
oneRes$fleets[[1]] <- setUnitsNA(oneRes$fleets[[1]])
oneRes_flt    <- fltSum(oneRes)
oneRes_fltStk <- fltStkSum(oneRes)
oneRes_mt     <- mtSum(oneRes)
oneRes_mtStk  <- mtStkSum(oneRes)
oneRes_adv    <- advSum(oneRes)

head(oneRes_bio)
head(oneRes_flt)
head(oneRes_fltStk)
head(oneRes_mt)
head(oneRes_mtStk)
head(oneRes_adv)

oneRes_bioQ    <- bioSumQ(oneRes_bio)
oneRes_fltQ    <- fltSumQ(oneRes_flt)
oneRes_fltStkQ <- fltStkSumQ(oneRes_fltStk)
oneRes_mtQ     <- mtSumQ(oneRes_mt)
oneRes_mtStkQ  <- mtStkSumQ(oneRes_mtStk)
```

```
oneRes_advQ    <- advSumQ(oneRes_adv)

head(oneRes_bioQ)
head(oneRes_fltQ)
head(oneRes_fltStkQ)
head(oneRes_mtQ)
head(oneRes_mtStkQ)
head(oneRes_advQ)

# Long format for a range of years

oneRes_bio    <- bioSum(oneRes, long = TRUE, years = ac(2016:2020))
oneRes_flt    <- fltSum(oneRes, long = TRUE, years = ac(2016:2020))
oneRes_fltStk <- fltStkSum(oneRes, long = TRUE, years = ac(2016:2020))
oneRes_mt     <- mtSum(oneRes, long = TRUE, years = ac(2016:2020))
oneRes_mtStk  <- mtStkSum(oneRes, long = TRUE, years = ac(2016:2020))
oneRes_adv    <- advSum(oneRes, long = TRUE, years = ac(2016:2020))

head(oneRes_bio)
head(oneRes_flt)
head(oneRes_fltStk)
head(oneRes_mt)
head(oneRes_mtStk)
head(oneRes_adv)

oneRes_bioQ    <- bioSumQ(oneRes_bio)
oneRes_fltQ    <- fltSumQ(oneRes_flt)
oneRes_fltStkQ <- fltStkSumQ(oneRes_fltStk)
oneRes_mtQ     <- mtSumQ(oneRes_mt)
oneRes_mtStkQ  <- mtStkSumQ(oneRes_mtStk)
oneRes_advQ    <- advSumQ(oneRes_adv)

head(oneRes_bio)
head(oneRes_flt)
head(oneRes_fltStk)
head(oneRes_mt)
head(oneRes_mtStk)
head(oneRes_adv)

# Wide format with seasonal disaggregation
# (Note: No seasonal disagregation available for adv summaries)

oneRes_bio    <- bioSum(oneRes, byyear = FALSE)
oneRes_flt    <- fltSum(oneRes, byyear = FALSE)
oneRes_fltStk <- fltStkSum(oneRes, byyear = FALSE)
oneRes_mt     <- mtSum(oneRes, byyear = FALSE)
oneRes_mtStk  <- mtStkSum(oneRes, byyear = FALSE)
oneRes_adv    <- advSum(oneRes) # Advice summary is only by year.

oneRes_bioQ    <- bioSumQ(oneRes_bio)
oneRes_fltQ    <- fltSumQ(oneRes_flt)
oneRes_fltStkQ <- fltStkSumQ(oneRes_fltStk)
oneRes_mtQ     <- mtSumQ(oneRes_mt)
```

```
oneRes_mtStkQ  <- mtStkSumQ(oneRes_mtStk)
oneRes_advQ    <- advSumQ(oneRes_adv)

# Long format and seasonal

oneRes_bio    <- bioSum(oneRes, long = TRUE)
oneRes_flt    <- fltSum(oneRes, long = TRUE, byyear = FALSE)
oneRes_fltStk <- fltStkSum(oneRes, long = TRUE, byyear = FALSE)
oneRes_mt     <- mtSum(oneRes, long = TRUE, byyear = FALSE)
oneRes_mtStk  <- mtStkSum(oneRes, long = TRUE, byyear = FALSE)
oneRes_adv    <- advSum(oneRes, long = TRUE) # Advice summary is only by year.

oneRes_bioQ    <- bioSumQ(oneRes_bio)
oneRes_fltQ    <- fltSumQ(oneRes_flt)
oneRes_fltStkQ <- fltStkSumQ(oneRes_fltStk)
oneRes_mtQ     <- mtSumQ(oneRes_mt)
oneRes_mtStkQ  <- mtStkSumQ(oneRes_mtStk)
oneRes_advQ    <- advSumQ(oneRes_adv)


#-----------------------------------------------
# Example OneIt: As one but with iterations.
#-----------------------------------------------

# Wide format (default)

oneItRes_bio    <- bioSum(oneItRes, scenario = 'with_iters')
oneItRes$fleets[[1]] <- setUnitsNA(oneItRes$fleets[[1]])
oneItRes_flt    <- fltSum(oneItRes, scenario = 'with_iters')
oneItRes_fltStk <- fltStkSum(oneItRes, scenario = 'with_iters')
oneItRes_mt     <- mtSum(oneItRes, scenario = 'with_iters')
oneItRes_mtStk  <- mtStkSum(oneItRes, scenario = 'with_iters')
oneItRes_adv    <- advSum(oneItRes, scenario = 'with_iters')

oneItRes_bioQ    <- bioSumQ(oneItRes_bio)
oneItRes_fltQ    <- fltSumQ(oneItRes_flt)
oneItRes_fltStkQ <- fltStkSumQ(oneItRes_fltStk)
oneItRes_mtQ     <- mtSumQ(oneItRes_mt)
oneItRes_mtStkQ  <- mtStkSumQ(oneItRes_mtStk)
oneItRes_advQ    <- advSumQ(oneItRes_adv)

# Long format for a range of years

oneItRes_bio    <- bioSum(oneItRes, long = TRUE, years = ac(2016:2020))
oneItRes_flt    <- fltSum(oneItRes, long = TRUE, years = ac(2016:2020))
oneItRes_fltStk <- fltStkSum(oneItRes, long = TRUE, years = ac(2016:2020))
oneItRes_mt     <- mtSum(oneItRes, long = TRUE, years = ac(2016:2020))
oneItRes_mtStk  <- mtStkSum(oneItRes, long = TRUE, years = ac(2016:2020))
oneItRes_adv    <- advSum(oneItRes, long = TRUE, years = ac(2016:2020))


oneItRes_bioQ    <- bioSumQ(oneItRes_bio)
oneItRes_fltQ    <- fltSumQ(oneItRes_flt)
```

```
oneItRes_fltStkQ <- fltStkSumQ(oneItRes_fltStk)
oneItRes_mtQ     <- mtSumQ(oneItRes_mt)
oneItRes_mtStkQ  <- mtStkSumQ(oneItRes_mtStk)
oneItRes_advQ    <- advSumQ(oneItRes_adv)

# Wide format with seasonal disaggregation
# (Note: No seasonal disagregation available for adv summaries)

oneItRes_bio    <- bioSum(oneItRes, byyear = FALSE)
oneItRes_flt    <- fltSum(oneItRes, byyear = FALSE)
oneItRes_fltStk <- fltStkSum(oneItRes, byyear = FALSE)
oneItRes_mt     <- mtSum(oneItRes, byyear = FALSE)
oneItRes_mtStk  <- mtStkSum(oneItRes, byyear = FALSE)
oneItRes_adv    <- advSum(oneItRes) # Advice summary is only by year.

oneItRes_bioQ    <- bioSumQ(oneItRes_bio)
oneItRes_fltQ    <- fltSumQ(oneItRes_flt)
oneItRes_fltStkQ <- fltStkSumQ(oneItRes_fltStk)
oneItRes_mtQ     <- mtSumQ(oneItRes_mt)
oneItRes_mtStkQ  <- mtStkSumQ(oneItRes_mtStk)
oneItRes_advQ    <- advSumQ(oneItRes_adv)

# Long format and seasonal

oneItRes_bio    <- bioSum(oneItRes, long = TRUE) # Biol summary is only by year.
oneItRes_flt    <- fltSum(oneItRes, long = TRUE, byyear = FALSE)
oneItRes_fltStk <- fltStkSum(oneItRes, long = TRUE, byyear = FALSE)
oneItRes_mt     <- mtSum(oneItRes, long = TRUE, byyear = FALSE)
oneItRes_mtStk  <- mtStkSum(oneItRes, long = TRUE, byyear = FALSE)
oneItRes_adv    <- advSum(oneItRes, long = TRUE) # Advice summary is only by year.

oneItRes_bioQ    <- bioSumQ(oneItRes_bio)
oneItRes_fltQ    <- fltSumQ(oneItRes_flt)
oneItRes_fltStkQ <- fltStkSumQ(oneItRes_fltStk)
oneItRes_mtQ     <- mtSumQ(oneItRes_mt)
oneItRes_mtStkQ  <- mtStkSumQ(oneItRes_mtStk)
oneItRes_advQ    <- advSumQ(oneItRes_adv)

oneItRes_risk <- riskSum( oneItRes, Bpa = c(stk1= 900), Blim = c(stk1 = 600),
                          Prflim = c(fl1 = 0), scenario = 'alternative')

oneItRes_npv  <- npv(oneItRes, y0 = '2014')


#-----------------------------------------------
# Example Multi: Two stock, two fleet, four iters.
#-----------------------------------------------

# Wide format (default)

multiRes_bio    <- bioSum(multiRes)
multiRes$fleets <- FLFleetsExt(lapply(multiRes$fleets, function(x) setUnitsNA(x)))
multiRes_flt    <- fltSum(multiRes)
```

```
multiRes_fltStk <- fltStkSum(multiRes)
multiRes_mt     <- mtSum(multiRes)
multiRes_mtStk  <- mtStkSum(multiRes)
multiRes_adv    <- advSum(multiRes)

multiRes_bioQ    <- bioSumQ(multiRes_bio)
multiRes_fltQ    <- fltSumQ(multiRes_flt)
multiRes_fltStkQ <- fltStkSumQ(multiRes_fltStk)
multiRes_mtQ     <- mtSumQ(multiRes_mt)
multiRes_mtStkQ  <- mtStkSumQ(multiRes_mtStk)
multiRes_advQ    <- advSumQ(multiRes_adv)

# Long format for a range of years

multiRes_bio    <- bioSum(multiRes, long = TRUE, years = ac(2016:2020))
multiRes_flt    <- fltSum(multiRes, long = TRUE, years = ac(2016:2020))
multiRes_fltStk <- fltStkSum(multiRes, long = TRUE, years = ac(2016:2020))
multiRes_mt     <- mtSum(multiRes, long = TRUE, years = ac(2016:2020))
multiRes_mtStk  <- mtStkSum(multiRes, long = TRUE, years = ac(2016:2020))
multiRes_adv    <- advSum(multiRes, long = TRUE, years = ac(2016:2020))

multiRes_bioQ    <- bioSumQ(multiRes_bio)
multiRes_fltQ    <- fltSumQ(multiRes_flt)
multiRes_fltStkQ <- fltStkSumQ(multiRes_fltStk)
multiRes_mtQ     <- mtSumQ(multiRes_mt)
multiRes_mtStkQ  <- mtStkSumQ(multiRes_mtStk)
multiRes_advQ    <- advSumQ(multiRes_adv)

# Wide format with seasonal disaggregation
# (Note: No seasonal disagregation available for adv summaries)

multiRes_bio    <- bioSum(multiRes, byyear = FALSE)
multiRes_flt    <- fltSum(multiRes, byyear = FALSE)
multiRes_fltStk <- fltStkSum(multiRes, byyear = FALSE)
multiRes_mt     <- mtSum(multiRes, byyear = FALSE)
multiRes_mtStk  <- mtStkSum(multiRes, byyear = FALSE)
multiRes_adv    <- advSum(multiRes) # Advice summary is only by year.

multiRes_bioQ    <- bioSumQ(multiRes_bio)
multiRes_fltQ    <- fltSumQ(multiRes_flt)
multiRes_fltStkQ <- fltStkSumQ(multiRes_fltStk)
multiRes_mtQ     <- mtSumQ(multiRes_mt)
multiRes_mtStkQ  <- mtStkSumQ(multiRes_mtStk)
multiRes_advQ    <- advSumQ(multiRes_adv)

# Long format and seasonal

multiRes_bio    <- bioSum(multiRes, long = TRUE, byyear = FALSE)
multiRes_flt    <- fltSum(multiRes, long = TRUE, byyear = FALSE)
multiRes_fltStk <- fltStkSum(multiRes, long = TRUE, byyear = FALSE)
multiRes_mt     <- mtSum(multiRes, long = TRUE, byyear = FALSE)
multiRes_mtStk  <- mtStkSum(multiRes, long = TRUE, byyear = FALSE)
multiRes_adv    <- advSum(multiRes, long = TRUE) # Advice summary is only by year.
```

```
multiRes_bioQ    <- bioSumQ(multiRes_bio)
multiRes_fltQ    <- fltSumQ(multiRes_flt)
multiRes_fltStkQ <- fltStkSumQ(multiRes_fltStk)
multiRes_mtQ     <- mtSumQ(multiRes_mt)
multiRes_mtStkQ  <- mtStkSumQ(multiRes_mtStk)
multiRes_advQ    <- advSumQ(multiRes_adv)

multiRes_npv  <- npv(multiRes, y0 = '2014')
risk_multiRes <- riskSum( multiRes, Bpa = c(stk1= 135000, stk2 = 124000),
                       Blim = c(stk1= 96000, stk2 = 89000), Prflim = c(fl1 = 0, fl2 = 0),
                          scenario = 'alternative')


## End(Not run)
```

---

brpsson *Reference points for a seasonal model*

Description

Function to estimate reference points when having a seasonal model

Usage

```
brpsson(
  stk,
  B0,
  R0,
  rec.ss = 1,
  ssb.ss = 1,
  sr_model,
  sr_params,
  Fprop = rep(1/dim(stk)[4], dim(stk)[4]),
  Fscan = seq(0, 4, by = 0.01),
  oldest = 100,
  nrun = 200,
  tol = 0.01
)
```

Arguments

stk	An FLStock object.
B0	The value of the virgin biomass.
R0	The expected recrutiment in the virgin population.
rec.ss	The recruitment season (numeric). Default value = 1.
ssb.ss	The spawning season (numeric). Default value = 1.

sr_model	A character with the name of the model to simulate the recruitment process.
sr_params	A named vector with the SR parameter values.
Fprop	A vector with the same length as the number of seasons with the proportion of F by season. By default the same proportion in all the seasons is assumed.
Fscan	A vector with the F values to be simulated. By default, will scan values between 0 and 4 (by 0.01 increments).
oldest	The maximum age class to be considered. Default 100.
nrun	The maximum number of years to project forward until equilibrium.
tol	The desired accuracy.

See Also

plotBRPsson

Examples

```
library(FLBEIA)
data(multistk) # object with 2 seasons and 3 iterations

stk <- trim( multistk, year=1, iter = 1)

# loop for different catch proportions by season
for (p in seq(0.1,0.9,0.1)) {
  fruns <- brpsson( stk, B0=1e+05, R0=27489766, rec.ss=2, ssb.ss=2,
                    sr_model="bevholt", sr_params=c( a = 29988835.109, b = 9090.909),
                    Fprop = c(p, 1-p))
}

plotBRPsson( fruns, pdfnm="stk_Fbar_vs_SPR.pdf")
```

---

calculate.q.sel.flrObjs

Calculates selectivity-related parameters

Description

Given stock abundances and catches (i.e. landings and discards), this function estimates values for `fleets[[fl]]@metiers[[mt]]@catches[[st]]@catch.q`, `fleets[[fl]]@metiers[[mt]]@catches[[st]]@landings.s` and `fleets[[fl]]@metiers[[mt]]@catches[[st]]@discards.sel` for all years except the simulation years. For the simulation years, the parameter values are set as the mean of the parameters along `mean.yrs`.

Usage

```
calculate.q.sel.flrObjs(biols, fleets, BDs, fleets.ctrl, mean.yrs, sim.yrs)
```

Arguments

biols	An FLBiols object.
fleets	An FLFleetsExt object. An extended version of the FLFleet object defined in FLCore.
BDs	A list of FLSRsim objects. One per biomass dynamic stock in biols object.
fleets.ctrl	Fleets' control file containing the catch production function for each fleet and stock.
mean.yrs	A character vector with the name of the years used to calculate mean selectivity.
sim.yrs	A character vector with the name of the years in the projection period.

Value

A FLFleetsExt object.

---

checkFLBEIAData *Checking inputs of FLBEIA function*

Description

This functions checks wether some conditions are met for the inputs in the different arguments of the FLBEIA function. There are also checking functions for specific inputs: biols, fleets, SRs, advice, and obs.ctrl arguments.

Functions available:

- `checkFLBEIAData`: for checking inputs for differents arguments of FLBEIA function (of class FLBiols).
- `checkBiols`: for checking biols argument of FLBEIA function (of class FLBiols).
- `checkFleets`: for checking fleets argument of FLBEIA function (of class FLFleetsExt).
- `checkSRs`: for checking SRs argument of FLBEIA function (list of FLSRsim objects).
- `checkAdvice`: for checking advice argument of FLBEIA function (of class list with two FLQuant elements, TAC and quota.share).
- `checkObsctrl`: for checking obs.ctrl argument of FLBEIA function (of class list).

Usage

```
checkFLBEIAData(
  biols,
  SRs = NULL,
  BDs = NULL,
  fleets,
  covars = NULL,
  indices = NULL,
  advice = NULL,
  main.ctrl,
```

```
  biols.ctrl,
  fleets.ctrl,
  covars.ctrl,
  obs.ctrl,
  assess.ctrl,
  advice.ctrl
)

checkBiols(object)

checkFleets(object, ctrl = NULL)

checkSRs(object)

checkBDs(object)

checkAdvice(object)

checkObsctrl(object)
```

Arguments

biols	An FLBiols object.
SRs	A list of FLSRsim objects.
BDs	A list of FLBDsim objects.
fleets	An FLFleetsExt object.
covars	A list of FLQuants.
indices	A list of FLIndices.
advice	A list with two FLQuant elements, TAC and quota.share.
main.ctrl	A list with the settings to control the main function.
biols.ctrl	A list with the settings to control the biological operating model for each stock.
fleets.ctrl	A list with the settings to control the fleets operating model for each fleet.
covars.ctrl	A list with the settings to control the covars operating model for each fleet.
obs.ctrl	A list with the settings to control the observation model for each stock.
assess.ctrl	A list with the settings to control the specify the assessment model for each stock.
advice.ctrl	A list with the settings to control the advice model for each stock.
object	An object of the appropiate class depending on the FLBEIA argument to be checked: • biols : A FLBiols object. • fleets :A FLFleetsExt object. • SRs : A list of FLSRsim objects. • BDs : A list of FLBDsim objects.

- advice : A list with two FLQuant elements, TAC and quota.share..
- obs.ctrl: A list.

ctrl — An object of class list (optional argument), input for fleets.ctrl argument of FLBEIA function.

Value

An error message if there is any error detected.

---

create.advice.ctrl *advice.ctrl object creator*

Description

It creates the advice.ctrl object to be used in the call to the main function FLBEIA.

Usage

```
create.advice.ctrl(stksnames, HCR.models = NULL, ...)
```

Arguments

stksnames	A vector with the name of the stocks in the OM.
HCR.models	A character vector of the same length as stksnames with the name of the HCR used to generate the management advice.
...	any extra arguments necessary in the HCR specific creators. '...' are extracted using 'list(...)', this generates a named list with the extra arguments. To assure the correct functioning the extra arguments must have a name.

Value

A list of lists with the basic structure of the advice.ctrl object.

---

create.advice.data *FLBEIA easy conditioning: advice argument creator*

Description

create.advice.data function creates a list (elements: TAC, TAE and quota.share)

Usage

```
create.advice.data(yrs, ns, ni, stks.data, fleets)
```

Arguments

yrs	A vector with c(first.yr,proj.yr, last.yr) where • first.yr: First year of simulation (number). • proj.yr: First year of projection (number). • last.yr: Last year of projection (number).
ns	Number of seasons (number).
ni	Number of iterations (number).
stks.data	A list with the names of the stocks and the following elements: Optionals: • stk_advice.TAC.flq: TAC of the stock 'stk' (FLQuant). • stk_advice.TAE.flq: TAE of the stock 'stk' (FLQuant). • stk_advice.quota.share.flq: Quota share of the stock 'stk' (FLQuant). • stk_advice.avg.yrs: Historic years to calculate the average of TAC, TAE or quota share of the stock 'stk' (FLQuant).
fleets	Optional argument only required if stk_advice.quota.share is not specified. It could be the output of create_fleets_FLBEIA function (FLFleets).

Value

A list with TAC, TAE and quota.share elements.

---

create.assess.ctrl *assess.ctrl object creator*

Description

It creates the assess.ctrl object to be used in the call to the main function FLBEIA.

Usage

```
create.assess.ctrl(stksnames, assess.models = NULL, assess.ctrls = NULL, ...)
```

Arguments

stksnames	A vector with the name of the stocks in the OM.
assess.models	A character vector of the same length as stksnames with the name of the model used to obtaine the perceived population in the MP.
assess.ctrls	A list of the same length as stksnames with the arguments needed to fit the assessment model.
...	any extra arguments necessary in the model specific creators. '...' are extracted using 'list(...)', this generates a named list with the extra arguments. To assure the correct functioning the extra arguments must have a name.

Value

A list of lists with the basic structure of the assess.ctrl object.

`create.BDs.data` *FLBEIA easy conditioning: BDs argument creator*

Description

create.BDs.data function creates a list of FLBDsim objects.

Usage

```
create.BDs.data(yrs, ns, ni, stks.data)
```

Arguments

`yrs`	A vector with c(first.yr,proj.yr, last.yr) where: • first.yr: First year of simulation (number). • proj.yr: First year of projection (number). • last.yr: Last year of projection (number).
`ns`	Number of seasons (number).
`ni`	Number of iterations (number).
`stks.data`	A list with the name of the stks and the following elements: • stk.unit: Number of units of the stock (number). • stk.age.min: Minimum age class of the stock (number). • stk.age.max: Maximum age class of the stock (number). • stk_bd.model: Name of the model to simulate biomass dinamics of the stock (character). • stk_params.name: Name of the parameters (vector). • stk_params.array: Parameter values (array). • stk_biomass.flq: Biomass values (FLQuant). • stk_catch.flq: Catch values (FLQuant). • stk_range.plusgroup: Plusgroup age (numeric). • stk_range.minyear: Minimum year (numeric). • stk_alpha: Maximum variability of carrying capacity. Optionals: • stk_gB.flq: Surplus production (FLQuant). • stk_uncertainty.flq: Uncertainty (FLQuant).

Value

A list of FLBDsim objects.

`create.biol.arrays` *Function to generate an FLBiol object given inputs as arrays*

Description

This function generates an FLBiol object, given the data inputs as arrays. Supported formats are Excel (xls and xlsx) and R format (RData).

Usage

```
create.biol.arrays(
  filename = NULL,
  data = NULL,
  name = NA,
  ages,
  hist.yrs,
  sim.yrs,
  fbar = NULL,
  mean.yrs,
  source = "rdata",
  unit = list()
)
```

Arguments

`filename`	A character vector with the name of the files containing the stock data. Supported formats are Excel (xls and xlsx) and R format (RData). In case of using R format, the information must be stored in `data` object (consisting in a list with the different elements). The following information is compulsory: abundances in numbers at age (n), mean weight at age (wt), maturity (mat), natural mortality (m), moment of the year when spawning occurs in percentage (spwn), fishing mortality at age (f) and catch in numbers at age (caa). For the rest of information, if not provided, default values are set. For example, fecundity (fec) is set to 1, landings and discard in numbers at age (laa and daa) are set to cca and 0, respectively. Finally for weights, if missing, weights at age for landings (wl) and discards (wd) are set to the weights in the population and weights at age for catch (wc) are set to the weighted mean of the weights of landings and discards.
`data`	An R object with the stock data.
`name`	A character (optional) with the name of the stock.
`ages`	A numeric vector with the age classes of stock.
`hist.yrs`	A vector with the historical years.
`sim.yrs`	A vector with the simulation years.
`fbar`	A numeric vector with the age range (min,max) to be used for estimating average fishing mortality.

mean.yrs	A vector with the years used to compute the mean to condition the parameters in the projection period.
source	Character, 'excel', 'rdata', 'FLStock' or 'object'. 'rdata' (default) if an RData object is used, 'excel' if the data is provided in an Excel file, 'FLStock' if the data is provided in and FLSTock object and 'object' if the data is an object of the working environment.
unit	A list with the units of the different elements included in filename. Unitless objects must be set to " or character(1). This parameter is only required if excel==FALSE. When using Excell files the units are taken from the first row and column (cell A1) of each sheet. If the cell is empty then units are set to NA, in case of an unitless object then 1 must be inputed into cell A1.

Value

An FLBiol.

Author(s)

Dorleta Garcia & Sonia Sanchez.

See Also

FLBiol, create.fleets.arrays

---

create.biols.ctrl *biols.ctrl object creator*

Description

It creates the biols.ctrl object to be used in the call to the main function FLBEIA.

Usage

```
create.biols.ctrl(stksnames, growth.models = NULL, immediate = FALSE, ...)
```

Arguments

stksnames	A vector with the name of the stocks in the OM.
growth.models	A character vector of the same length as stksnames with the name of the model used to project the stock populations in the simulation.
immediate	logical, indicating if the warnings should be output immediately.
...	any extra arguments necessary in the model specific creators. '...' are extracted using 'list(...)', this generates a named list with the extra arguments. To assure the correct functioning the extra arguments must have a name.

Value

A list of lists with the basic structure of the biols.ctrl object.

`create.biols.data` *FLBEIA easy conditioning: biols argument creator*

Description

create.biols.data function creates an FLBiols object.

Usage

```
create.biols.data(yrs, ns, ni, stks.data)
```

Arguments

`yrs`	A vector with c(first.yr,proj.yr, last.yr) where • first.yr: First year of simulation (number). • proj.yr: First year of projection (number). • last.yr: Last year of projection (number).
`ns`	Number of seasons (number).
`ni`	Number of iterations (number).
`stks.data`	A list with the name of the stks and the following elements: • stk.unit: Number of units of the stock (number). • stk.age.min: Minimum age class of the stock (number). • stk.age.max: Maximum age class of the stock (number). • stk_n.flq: Numbers at age in the population(FLQuant). • stk_wt.flq: Weight at age of an individual (FLQuant). • stk_m.flq: Mortality rate at age of the population (FLQuant). • stk_fec.flq: Fecundity at age (FLQuant). • stk_mat.flq: Percentage of mature individuals at age (FLQuant). • stk_spwn.flq: Proportion of time step at which spawning ocurrs (FLQuant). • stk.range.plusgroup: Plusgroup age (number). • stk.range.minyear: Minimum year (number). • stk.range.maxyear: Maximum year (number). • stk_range.minfbar: Minimum age to calculate average fishing mortality (number). • stk_range.maxfbar: Maximum age to calculate average fishing mortality (number). • stk_biol.proj.avg.yrs: Historic years to calculate the average of spwn, fec, m and wt for the projection (vector).

Value

An FLBiol object

`create.covars.ctrl` *covars.ctrl object creator*

Description

It creates the covars.ctrl object to be used in the call to the main function FLBEIA.

Usage

```
create.covars.ctrl(
  cvrsnames,
  process.models = NULL,
  flq,
  immediate = FALSE,
  ...
)
```

Arguments

`cvrsnames`	A vector with the name of the covariates in the OM.
`process.models`	A character vector of the same length as cvrsnames with the name of the process model followed by each of the covariates. The first element corresponds with the process model of the first covariable in cvrsnames, the second with the second and so on. The default is NULL in which case 'fixedCovar' is used for **all** the covariates.
`flq`	An FLQuant to give structure to the FLQuants to be used within the function, the dimension and dimnames in 'year', 'season' and 'iter' will be used to create the necessary FLQuants.
`immediate`	logical, indicating if the warnings should be output immediately.
`...`	any extra arguments necessary in the HCR specific creators. '...' are extracted using 'list(...)', this generates a named list with the extra arguments. To assure the correct functioning the extra arguments must have a name.

Value

A list of lists with the basic structure of the covars.ctrl object.

`create.ecoData` *Function to generate the covar object neccesary to compute economic indicators and fills in economic slots in FLFleets object*

Description

This function generates an covar object with the economic indicators and fills in vcost, fcost, capacity and crewshare slots in FLFleets object. The data is given in excel File (xls and xlsx).

Usage

```
create.ecoData(file, fltObj, hist.yrs, mean.yrs, sim.yrs)
```

Arguments

`file`	An excel file with the economic data. The names used for the sheets and the columns must be the same used to name fleets and metiers in fltObj. However, the fleets in the excel file can be a subset of the fleets in the fleetsObj, i.e., is not neccesary to provide economic data for all the fleets. The order of metiers in the columns data must correspond with the name of the metiers used in the FLFleet object.
`fltObj`	An FLFleets object with the structure of the fleet and which may contain historical data.
`hist.yrs`	A vector with the historical years.
`mean.yrs`	A vector with the years used to compute the mean to condition the parameters in the projection period.
`sim.yrs`	A vector with the simulation years.

Value

An /codeFLFleetsExt.

Author(s)

Dorleta Garcia & Sonia Sanchez.

See Also

`FLFleetsExt`, `create.biol.arrays`

`create.fleets.arrays` *Function to generate an FLFleetsExt object given inputs as arrays*

Description

This function generates an FLFleetsExt object, given the data inputs as arrays. Supported formats are Excel (xls and xlsx) and R format (RData).

Usage

```
create.fleets.arrays(
  stk_objs,
  caa_objs,
  caa_objs_path,
  price_objs,
  price_objs_path,
  catch_obj,
  effort_obj,
  flt_obj = NULL,
  stk_nms = NA,
  flt_nms,
  flt_mt_nms,
  flt_mt_stk_nms,
  ages = NULL,
  hist.yrs,
  sim.yrs,
  mean.yrs,
  new_hist.yrs = hist.yrs,
  update_catch_effort = TRUE,
  update_price = TRUE,
  update_weight = TRUE,
  caa_flt_mt_correspondences = NULL,
  paa_flt_mt_correspondences = NULL,
  caaOpt,
  priceOpt,
  excel = TRUE
)
```

Arguments

`stk_objs`	A character vector with the names of the files containing the stocks data. See create.biol.arrays for more detail. Supported format is only Excel (xls and xlsx), each stock can be in different format.
`caa_objs`	A character vector with the names of the files containing the catch at age data (in numbers), both for landings and discards. Supported formats are Excel (xls and xlsx) and R format (RData), each file can be in different format. The number of required files depend of the value of `caaOpt` argument:

- caaOpt = 1 or 2 => one per stock and fleet. Named vector stknm_fltnm
- caaOpt = 3, 4 or 5=> one per stock. Named vector stknm

If NULL, the function looks for "caa_stknm_flnm.xlsx" in caaOpt = 1 or 2 and "caa_stknm.xlsx" in caaOpt = 3 ,4 or 5.

caa_objs_path A character vector with the caa_objs file path.

price_objs A character vector with the names of the files containing the price at age data. Supported formats are Excel (xls and xlsx) and R format (RData), each prices file can be in different format. The number of required files depend of the value of caaOpt argument:

- caaOpt = 1 or 2 => one per stock and fleet. Named vector stknm_fltnm.
- caaOpt = 3, 4 or 5=> one per stock. Named vector stknm.

price_objs_path
A character vector with the price_objs file path.

catch_obj A character vector with the names of the files containing the catch data in Fcube format. Supported formats are Excel (xls and xlsx) and R format (RData) and required columns are 'year', 'fleet', 'metier', 'stock', 'category' and 'catch'.

effort_obj A character vector with the names of the files containing the effort data in Fcube format. Supported formats are Excel (xls and xlsx) and R format (RData) and required columns are 'year', 'fleet', 'metier' and 'effort'. Both catch_obj and effort_obj must be in the same format.

flt_obj An FLFleets object (optional) with the structure of the fleet and which may contain historical data. If this object is provided, then the arguments stk_nms, flt_nms, flt_mt_nms and ages will not be used and excel argument must be set to FALSE.

stk_nms A character vector (optional) with the name of all the stocks caugth by the different fleets.

flt_nms A character vector with the name of the fleets.

flt_mt_nms A list with one element per fleet. In turn, each element is a character vector with the names of the metiers in the corresponding fleet.

flt_mt_stk_nms A list with one element per fleet and metier. In turn, each element is a character vector with the names of the stocks in the corresponding fleet and metier.

ages A list with one element per stock, with the age classes of the stock.

hist.yrs A vector with the historical years.

sim.yrs A vector with the simulation years.

mean.yrs A vector with the years used to compute the mean to condition the parameters in the projection period.

new_hist.yrs A vector with the years from input files that will be used to condition the parameters in the historic years. If a value is not provided, the it is set equal to hist.yrs.

update_catch_effort
Logical. If TRUE (default), catch and effort must be provided and catch_obj, effort_obj arguments are required.

update_price Logical. If TRUE (default), prices must be provided and price_objs and price_objs_path arguments are required.

update_weight Logical. If TRUE (default), weights at age for landings and discards must be provided, so wl and wd sheets are required in files listed in stk_objs argument.

caa_flt_mt_correspondences
An Excel file name. This file must contain one sheet per stock, with the correspondences between the fleet segments used in caaa_obj data and the fleet metier segmentation used in the analysis. If the file does not exist, it is supposed that the caa data is given by fleet and metier.

paa_flt_mt_correspondences
An Excel file name. This file must contain information on prices correspondences, with same format and requirements as caa_flt_mt_correspondences argument.

caaOpt A code number to determine the way in wich catch at age data are provided. The option to be used depends on the data availabiltiy, from data rich to data-poor and the following codes are available:

- 1If catch at age data is available at metier level for all the metiers.
- 2If catch at age data is only available at fleet level.
- 3If catch at age data is disaggregated but the segments do not correspond exactly with the metiers/fleets considered in the case study.
- 4If catch at age data is only available at stock level.
- 5If we want to use the data available previously in the FLCatch objects from flt_obj to derive catch profiles at age and then apply caaOpt==3 using only one fleet segment, fseg, which represents all the fleets and metiers. Note: This approach could lead to a different total catch at age profile derived from the fleets to those in the stocks.

priceOpt A code number to determine the way in wich price at age data are provided. The option to be used depends on the data availabiltiy, from data rich to data-poor and the following codes are available:

- 1If price data is available at metier level for all the metiers.
- 2If price data is only available at fleet level.
- 3If price data is disaggregated but the segments do not correspond exactly with the metiers/fleets considered in the case study.
- 4If price data is only available at stock level.

excel Logical. If TRUE (default), the data in the Excel file is used to create the stucture of the FLFleets object; whereas if FALSE, the flt_obj object is used instead.

Value

An FLFleetsExt.

Author(s)

Dorleta Garcia & Sonia Sanchez.

See Also

FLFleetsExt, create.biol.arrays

`create.fleets.ctrl` *fleets.ctrl object creator*

Description

It creates the fleets.ctrl object to be used in the call to the main function FLBEIA.

Usage

```
create.fleets.ctrl(
  fls,
  n.fls.stks,
  fls.stksnames,
  catch.threshold = NULL,
  seasonal.share = NULL,
  effort.models = NULL,
  capital.models = NULL,
  catch.models = NULL,
  price.models = NULL,
  flq,
  ...
)
```

Arguments

`fls`	character vector with fleet names
`n.fls.stks`	numeric vector with the same length as fls with the declaration of the number of stocks caugth by each of the fleets.
`fls.stksnames`	character vector with length = sum(n.fls.stks), with the names of the stocks caught by the fleet, the vector must follow the order used in the previous argument. • the first n.fls.stks[1] elements correspond to the stocks caught by the first fleet in fls • the following n.fls.stks[2] elements correspond to the stocks caught by the second fleet in fls • and so on.
`catch.threshold`	if(NULL) => 0.9 for all the stocks (NULL is the default) else it must be an FLQuant with dim = c(nstks,ny,1,ns,nit)
`seasonal.share`	an FLQuant with dimension [num. fleets, num. years, 1,num. seasons, 1, num. iterations] with elements between 0 and 1 to indicate how the quota of each fleet is distributed along seasons. The sum along seasons (seasonSums) must return an FLQuant with all elements equal to 1.

`effort.models` characted vector with the same length as fls with the effort model followed by each of the fleet. the first element correspond with the effort model of the first fleet in fls, the second with the second and so on. The default is NULL in which case 'fixedEffort' is used for **all** the fleets.

`capital.models` characted vector with the same length as fls with the capital model followed by each of the fleet. the first element correspond with the capital model of the first fleet in fls, the second with the second and so on. The default is NULL in which case 'fixedCapital' is used for **all** the fleets.

`catch.models` characted vector with the same length as sum(n.fls.stks) with the catch model followed by each of the fleet for each stock. the first element correspond with the catch model of the first fleet in fls and the first stock in fls.stksnames, the second with the second and so on. The default is NULL in which case 'Cobb-DouglasAge' is used for **all** the fleets.

`price.models` characted vector with the same length as sum(n.fls.stks) with the price model followed by each of the fleet for each stock. the first element correspond with the price model of the first fleet in fls and the first stock in fls.stksnames, the second with the second and so on. The default is NULL in which case 'fixedPrice' is used for **all** the fleets.

`flq` An FLQuant to give structure to the FLQuants to be used within the function, the dimension and dimnames in 'year', 'season' and 'iter' will be used to create the necessary FLQuants.

`...` Any extra arguments necessary in the model specific creators. '...' are extracted using 'list(...)', this generates a named list with the extra arguments. To assure the correct functioning the extra arguments must have a name, for example, elas = FLQuant(1,dimnames = DimsNms).

Value

A list of lists with the basic structure of the fleets.ctrl object.

`create.fleets.data` *FLBEIA easy conditioning: fleets argument creator*

Description

create.fleets.data function creates an FLFleetsExt object

Usage

```
create.fleets.data(yrs, ns, ni, fls.data, stks.data)
```

Arguments

`yrs` A vector with c(first.yr,proj.yr, last.yr) where

- first.yr: First year of simulation (number).
- proj.yr: First year of projection (number).
- last.yr: Last year of projection (number).

`ns` Number of seasons (number).

`ni` Number of iterations (number).

`fls.data` A list with the name of the fleets and the following elements:

- fl.met: Name of the metiers in the fleet 'fl' (vector).
- fl.met.stks: Name of the stocks in the metier 'met' and fleet 'fl' (vector).
- fl_effort.flq: 'fl' fleet's effort (FLQuant).
- fl.met_effshare.flq: 'fl' fleet and 'met' metier's effort share (FLQuant).
- fl.met.stk_landings.n.flq: 'fl' fleet,'met' metier and 'stk' stock's landings in numbers at age.
- fl_proj.avg.yrs: Historic years to calculate the average of effort, fcost, crewshare, capacity in 'fl' fleet (vector) in the projection years.

Optionals:

- fl_capacity.flq: 'fl' fleet's capacity (FLQuant).
- fl_fcost.flq: 'fl' fleet's fixed cost (FLQuant).
- fl_crewshare.flq: 'fl' fleet's crewshare (FLQuant).
- fl.met_vcost.flq: 'fl' fleet and 'met' metier's variable costs (FLQuant).
- fl.met.stk_landings.wt.flq: 'fl' fleet,'met' metier and 'stk' stock's mean weight of landings at age.
- fl.met.stk_discards.n.flq: 'fl' fleet,'met' metier and 'stk' stock's discards in numbers at age (FLQuant).
- fl.met.stk_discards.wt.flq: 'fl' fleet,'met' metier and 'stk' stock's mean weight of discards at age.
- fl.met.stk_price.flq: 'fl' fleet,'met' metier and 'stk' stock's price at age (FLQuant).
- fl.met.stk_alpha.flq: 'fl' fleet,'met' metier and 'stk' stock's Cobb Douglass alpha parameter (FLQuant).
- fl.met.stk_beta.flq: 'fl' fleet,'met' metier and 'stk' stock's Cobb Douglass beta parameter (FLQuant).
- fl.met.stk_catch.q.flq: 'fl' fleet,'met' metier and 'stk' stock's Cobb Douglass catch.q parameter (FLQuant).
- fl.met_proj.avg.yrs: Historic years to calculate the average of effshare,vcost for 'fl' fleet and 'met' metier in the projection period (vector).
- fl.met.stk_proj.avg.yrs: Historic years to calculate the average of landings.wt, discards.wt, landings.sel, discards.sel alpha,beta,catch.q for 'fl' fleet, 'met' metier and 'stk' stock in the projection years (vector).

`stks.data` A list with the name of the stocks and with the next elements:

- stk.unit: Number of units of the stock (number).

- stk.age.min: Minimum age of the stock (number).
- stk.age.max: Maximum age of the stock (number).

Optionals:

- stk_wt.flq: Mean weight at age of an individual (FLQuant). Required if fl.met.stk_landings.wt.flq is not defined.
- stk_n.flq: Numbers at age in the population(FLQuant). Required if Cobb Douglas parameters are not defined.
- stk_gB.flq: Biomass growth for the stock modeled in biomass (FLQuant). Required if Cobb Douglas parameters are not defined.

Value

An FLFleetsExt object.

`create.indices.data` *FLBEIA easy conditioning: indices argument creator*

Description

create.indices.data function creates an FLIndices object

Usage

```
create.indices.data(yrs, ns, ni, stks.data)
```

Arguments

`yrs` A vector with c(first.yr,proj.yr, last.yr) where

- first.yr: First year of simulation (number).
- proj.yr: First year of projection (number).
- last.yr: Last year of projection (number).

`ns` Number of seasons (number).

`ni` Number of iterations (number).

`stks.data` A list with the names of the stocks with indices and the following elements:

- stk.unit: Number of units of the stock (number).
- stk.age.min: Minimum age class of the stock (number).
- stk.age.max: Maximum age class of the stock (number).
- stk_indices: Name of indices for the stock 'stk' (vector).
- stk_ind_index.flq: Historical index data for index 'ind' of stock 'stk' (FLQuant).

Optionals:

- stk_ind_type: Type of index (character).
- stk_ind_distribution: Name of the statistical distribution of the 'ind' index values for stock 'stk' (character).

- stk_ind_index.var.flq: Variability in 'ind' index of stock 'stk' (FLQuant).
- stk_ind_index.q.flq: Catchability for 'ind' index of stock 'stk' (FLQuant).
- stk_ind_catch.n.flq: Catch at age in numbers for 'ind' index of stock 'stk'(FLQuant).
- stk_ind_catch.wt.flq: Mean weight at age in the catch for 'ind' index of stock 'stk' (FLQuant).
- stk_ind_effort.flq: Effort for 'ind' index of stock 'stk' (FLQuant).
- stk_ind_sel.pattern.flq: Selection pattern for 'ind' index of stock 'stk' (FLQuant).
- stk_ind_range.min: Minimum age in 'ind' index of stock 'stk' (number).
- stk_ind_range.max: Maximum age in 'ind' index of stock 'stk' (number).
- stk_ind_range.plusgroup: Plusgroup age in 'ind' index of stock 'stk' (number).
- stk_ind_range.minyear: First year with 'ind' index data of stock 'stk' (number).
- stk_ind_range.maxyear: Last year with 'ind' index data of stock 'stk' (number).
- stk_ind_range.startf: Minimum age for calculating average fishing mortality for 'ind' index of stock 'stk' (number).
- stk_ind_range.endf: Maximum age for calculating average fishing mortality for 'ind' index of stock 'stk' (number).

Value

An FLIndices object.

---

create.obs.ctrl *obs.ctrl object creator*

Description

It creates the obs.ctrl object to be used in the call to the main function FLBEIA.

Usage

```
create.obs.ctrl(
  stksnames,
  n.stks.inds = NULL,
  stks.indsnames = NULL,
  stkObs.models = NULL,
  indObs.models = NULL,
  immediate = FALSE,
  ...
)
```

Arguments

stksnames	A vector with the name of the stocks in the OM.
n.stks.inds	numeric vector with the same length as stksnames with the declaration of the number of abundance indices per stock, the order must be the same used in stksnames.
stks.indsnames	NULL or a character vector of length equal to sum(n.stks.inds) with the names of the indices per stock. The order must be the same used in the previous arguments.
stkObs.models	A character vector of the same length as stksnames with the name of the model used to observed stock data.
indObs.models	A character vector of the same length as stks.indsnames with the name of the model used to generate the abundance indices.
immediate	logical, indicating if the warnings should be output immediately.
...	any extra arguments necessary in the model specific creators. '...' are extracted using 'list(...)', this generates a named list with the extra arguments. To assure the correct functioning the extra arguments must have a name.

Value

A list of lists with the basic structure of the obs.ctrl object.

---

create.SRs.data *FLBEIA easy conditioning: SRs argument creator*

Description

create.BDs.data function creates a list of FLBDsim objects

Usage

```
create.SRs.data(yrs, ns, ni, stks.data)
```

Arguments

yrs	A vector with c(first.yr,proj.yr, last.yr) where: • first.yr: First year of simulation (number). • proj.yr: First year of projection (number). • last.yr: Last year of projection (number).
ns	Number of seasons (number).
ni	Number of iterations (number).
stks.data	A list with the name of the stks and the following elements: • stk.unit: Number of units of the stock (number). • stk.age.min: Minimum age class of the stock (number).

- stk.age.max: Maximum age class of the stock (number).
- stk_sr.model: Name of the model to simulate recruitment (character).
- stk_params.n: Number of parameters (number).
- stk_params.name: Name of the parameters (vector).
- stk_params.array: Parameter values (array).
- stk_rec.flq: Recruitment values (FLQuant).
- stk_ssb.flq: Spawning stock biomass values (FLQuant).
- stk_proportion.flq: Recruitment distribution in each time step as a proportion (FLQuant, values between 0 and 1).
- stk_prop.avg.yrs: Historical years to calculate the proportion average (vector).
- stk_timelag.matrix: Timelag between the spawning an recruitment (matrix [2, number of seasons]). For details see FLSRsim.
- stk_range.plusgroup: Plusgroup age (number).
- stk_range.minyear: Minimum year (number).

Optionals:

- stk_uncertainty.flq: Uncertainty (FLQuant).

Value

A list of FLSRsim objects.

`datasets` *FLBEIA datasets*

Description

Example datasets for the classes defined in FLBEIA.

Details

- `one`: A dataset for running FLBEIA. Example with one stock (age-structured), one fleet, annual steps (one season) and one iteration.
  - `oneBio` (`FLBiols`) : Biological information on the stock. In this case, the stock is age-structured.
  - `oneSR` (`FLSRsim`) : Stock-recruitment model for the stock in `oneBio` object.
  - `oneFl` (`FLFleetsExt`) : Information on the fleet and the metier considered.
  - `oneCv` (`list` of `FLQuants`) : Covariates information. In this case all are economic indicators.
  - `oneIndAge` and `oneIndBio` (`list` of `FLIndices`) : Indices, if avalable, for the different stocks in `oneBio`. Where `oneIndAge` and `oneIndBio` are indices with estimates in numbers at age and total biomass, respectively.
  - `oneAdv` (`list`) : Information on TAC and quota share.
  - `oneMainC` (`list`) : Settings to control the main function `FLBEIA`. The simulation years.

  - oneBioC (list) : Settings to control the biological operating model for the stock in oneBio object.
  - oneFlC (list) : Settings to control the fleet operating model for the fleet in oneFl object.
  - oneCvC (list) : Settings to control the covar operating model for each covariate in covars object. In this case all the covariates are fixed.
  - oneObsC, oneObsCIndAge and oneObsCIndBio (list) : Settings to control the observation model for the stock in oneBio object. In oneObsC the stock is observed without error and there are no indices available. Alternative control settings are available for cases when indices are observed, oneObsCIndAge and oneObsCIndBio.
  - oneAssC (list) : Settings to control the assessment model for each stock in oneBio object. In this case, no assessment is carried out.
  - oneAdvC (list) : Settings to control the advice model for the stock in oneBio object.
- oneIt: A dataset for running FLBEIA. Same as one dataset, but with three iterations.
- multi: A dataset for running FLBEIA. Example with two stocks (one age-structured and the other in biomass), two fleets (with 2 metiers each), four seasons and one iteration.
  - multiBio (FLBiols) : Biological information on the stocks (one is age-structured and the other one in total biomass).
  - multiSR (FLSRsim) : Stock-recruitment models for the age-structured stock in multiBio object. In this case a Beverton-Holt (bevholt) is selected.
  - multiBD (FLSRsim) : Biomass dynamic model for the stock in biomass in multiBio object. In this case Pella-Tomlinson model (PellaTom) is selected.
  - multiFlC (FLFleetsExt) : Information on the fleets and metiers. In this case there are two fleets, each one with two metiers, all of them capturing both stocks in multiBio object.
  - multiCv (list of FLQuants) : Covariates information. In this case all are economic indicators.
  - multiAdv (list) : Information on TAC and quota share.
  - multiMainC (list) : Settings to control the main function FLBEIA.
  - multiBioC (list) : Settings to control the biological operating model for each stock in multiBio object.
  - multiFlC (list) : Settings to control the fleet operating model for each fleet in fleets object.
  - multiCvC (list) : Settings to control the covar operating model for each covariate in covars object. In this case all the covariates are fixed.
  - multiObsC (list) : Settings to control the observation model for each stock in multiBio object. In this case the stock is observed without error and there are no indices available.
  - multiAssC (list) : Settings to control the assessment model for each stock in multiBio object. In this case, no assessment is carried out.
  - multiAdvC (list) : Settings to control the advice model for each stock in multiBio object.
- res_flbeia: A dataset with the outputs of FLBEIA runs given as input the different datasets (one, oneIt and multi).
  - oneRes (list) : Output of the FLBEIA function, given as input the data in the one dataset.

  - onIteRes (list) : Output of the FLBEIA function, given as input the data in the oneIt dataset.
  - multiRes (list) : Output of the FLBEIA function, given as input the data in the multi dataset.

- mur: A dataset for Stripped Red Mullet in the Bay of Biscay. Information on catch and abundance indices from Evohe survey.
  - catch (data.frame) : The total catch time series data by area from WGBIE report (ICES, 2017). Total catch data is available since 1975. In 1999 France did not report any data. As France is the main contributor to the total catch, the 1999 catch data was not included in the analysis.
  - evhoe (data.frame) : EVHOE abundance index time series, provided by Ifremer. The abundance index is available since 1997 and provides an estimation of the biomass together with a coefficient of variation.
- multistk: A dataset with an FLStock with multiple dimensions.
  - multistk (FLStock) : FLStock object with 2 seasons and 3 iterations.

Datasets can be loaded by issuing the data command, like in: data(one).

All available datasets can be checked by: data(package='FLBEIA').

References

ICES, 2017

See Also

FLBEIA, FLBiols, FLFleetsExt, FLSRsim, FLIndices, FLQuant

Examples

```
data(one)
data(res_flbeia)
```

---

ecoSum_damara *Biological summary functions*

Description

These functions return the biomass (B), fishing mortality (F), spawning stock biomass (SSB), recruitment (R), catches (C), landings (L) and discards (D) indicators. Also indicators comparing the reference points with the actual values, Bpa, Blim, Btarget, Fpa, Flim and Ftarget, so the biomass indicators are TRUE if the biomass is above them, and the fishing mortality indicators are TRUE if the fishing mortality is below them. ssb2Btarget and f2Ftarget return the ratio between SSB and F and the target reference point.

Usage

```
ecoSum_damara(fleets, flnms = "all", years, covars = NULL)

F_flbeia(obj, years = dimnames(obj$biols[[1]]@n)$year)

SSB_flbeia(obj, years = dimnames(obj$biols[[1]]@n)$year)

B_flbeia(obj, years = dimnames(obj$biols[[1]]@n)$year)

R_flbeia(obj, years = dimnames(obj$biols[[1]]@n)$year)

C_flbeia(obj, years = dimnames(obj$biols[[1]]@n)$year)

L_flbeia(obj, years = dimnames(obj$biols[[1]]@n)$year)

D_flbeia(obj, years = dimnames(obj$biols[[1]]@n)$year)

summary_flbeia(obj, years = dimnames(obj$biols[[1]]@n)$year)
```

Arguments

fleets	A FLFleetsExt object.
flnms	Names of the fleet for which the indicators will be calculated.
years	The years for which the indicators are extracted.
covars	List of FLQuants with information on covariates.
obj	The output of the FLBEIA function.

Details

- B_flbeia this function computes SSB.
- F_flbeia this function computes fishing mortiality.
- SSB_flbeia this function computes spawning stock biomass by species.
- R_flbeia this function computes recruitment by stock. If the stock is defined by age this function the recruiment is computed. ; If the stock is follows a biomass dynamics, this function gives the growth.
- C_flbeia this function computes catches by fleets and stock.
- L_flbeia this function computes landings by fleets and stock.
- D_flbeia this function computes the discards by fleets and stock.
- summary_flbeia this function computes recruitment, SSB, fishing mortality, total biomass for all stocks; and catch and landings for all fleets by year and stock.

Value

B_flbeia, F_flbeia... return an array with three dimensions (stock, year and iter). The summary_flbeia function returns an array with 4 dimensions (stock, year, iter, indicator) with the value of all the indicators.

extractBRP *Extracts reference points (specifically Bpa, Blim, Bmsy, Fpa, Flim and Fmsy) from advice.ctrl object, If not available, then values are set to NA.*

Description

The output can be used for the brp argument of bioSum function.

Usage

```
extractBRP(advice.ctrl, stkn, Btarget = NULL, Ftarget = NULL)
```

Arguments

advice.ctrl	A list with advice controls as the FLBEIA function argument advice.ctrl.
stkn	Names of the stocks.
Btarget	Named vector with the name of the target biological reference point for each element in stkn. Default "Bmsy".
Ftarget	Named vector with the name of the target fishing mortality reference point for each element in stkn. Default "Fmsy".

Value

A data frame with columns stock, iter and one colum per reference point with the value of the biological reference points per stock and iteration. The used reference points are Bpa, Blim, Bmsy, Fpa, Flim and Fmsy.

See Also

bioSum

Examples

```
## Not run:

library(FLBEIA)

data(one)
extractBRP(oneAdvC, stkn = names(oneBio))

data(oneIt)
extractBRP(oneItAdvC, stkn = names(oneItBio))

data(multi)
extractBRP(multiAdvC, stkn = names(multiBio))

# setting targets different to Bmsy and Fmsy
```

```
extractBRP(oneAdvC, stkn = names(oneBio),
           Btarget=setNames("Btrigger","stk1"), Ftarget=setNames("Ftrigger","stk1"))

Btarget <- setNames(c("Btrigger","Bmsy"),names(oneBio))
Ftarget <- setNames(c("Ftrigger","Fmsy"),names(oneBio))

extractBRP(multiAdvC, stkn = names(multiBio))


## End(Not run)
```

FLBDsim *FLBDsim class and methods*

Description

A class to simulate the growth of populations aggregated in biomass.

Arguments

...	Empty or FLQuants for 'biomass', 'catch' and 'uncertainty' slots and optionally the values for the rest of the slots.

Details

The FLBDsim has the following slots:

Value

An object of class FLBDsim.

Slots

`name` The name of the stock.

`desc` A description of the object.

`range` The range of the object.

`biomass` An FLQuant to store the biomass of the stock.

`gB` An FLQuant to store the surplus production of the stock.

`catch` An FLQuant to store the catch of the stock.

`uncertainty` An FLQuant to store the uncertainty that is multiplied to the biomass in every step of the simulation.

`covar` An FLQuants to store the covariates that are part of the growth model.

`model` A character with the name of the model to simulate the recruitment process.

`params` An array with dimension [numb.params, numb.year, numd.season, numb.iteration] with year and season and iteration dependent parameters of the growth model.

`alpha` An array with dimension [numb.year, numd.season, numb.iteration] with year, season and iteration dependent value bigger than one which indicates, in percentage, how big can be the biomass in comparison with the carrying capacity.

`name` The name of the object.

`desc` Character with the description of the object.

`range` A numeric vector with the range of the object as in other FLR objects.

`FLBEIA` *Run the FLBEIA bio-economic simulation model*

Description

FLBEIA is a simulation model that describes a fishery system under a Management Strategy Evaluation framework.The objective of the model is to facilitate the Bio-Economic evaluation of Management strategies. The model is multistock, multifleet and seasonal. The simulation is divided in 2 main blocks, the Operating Model (OM) and the Management Procedure (MP). In turn, the OM is divided in 3 components, the biological, the fleets and the covariables component. The MP is also divided in 3 components, the observation, the assessment and the advice.

Usage

```
FLBEIA(
  biols,
  SRs = NULL,
  BDs = NULL,
  fleets,
  covars = NULL,
  indices = NULL,
  advice = NULL,
  main.ctrl,
  biols.ctrl,
  fleets.ctrl,
  covars.ctrl,
  obs.ctrl,
  assess.ctrl,
  advice.ctrl
)
```

Arguments

`biols`	An FLBiols object.
`SRs`	A list of FLSRsim objects. One per age structured stock in biols object.
`BDs`	A list of FLBDsim objects. One per biomass dynamic stock in biols object.
`fleets`	An FLFleetsExt object. An extended version of the FLFleet object defined in FLCore.

covars	A list of FLQuants used to store any kind of variables that are used within the simulation and are not stored in the standard objects.
indices	A list of FLIndices. Each element must correspond with one of the stocks in biols object.
advice	A list with two FLQuant elements, TAC and quota.share. TAC is an FLQuant with quant dimension equal to the number of stocks in biols object, the names used in in the quant dimension must be equal to those used in biols. quota.share is a list with one element per stock in biols object indicating the quota share per stock and fleet. The quant dimension of the elements must be equal to the number of fleets and the names used must be equal to those in fleets objects.
main.ctrl	A list with the settings to control the main function (the year range,...).
biols.ctrl	A list with the settings to control the biological operating model for each stock (the population dynamic model used, additional parameters,...)
fleets.ctrl	A list with the settings to control the fleets operating model for each fleet (the fleets' short and long term dynamic models used, price model, additional parameters,...)
covars.ctrl	A list with the settings to control the covars operating model for each fleet (a dynamic model for each covariable, additional parameters, ...)
obs.ctrl	A list with the settings to control the observation model for each stock (the observation model for the stock, for stock dependent indices, additional parameters, ...)
assess.ctrl	A list with the settings to control the specify the assessment model for each stock (the assessment model for the stock and the control parameters used to run the model)
advice.ctrl	A list with the settings to control the advice model for each stock (the HCR for each stock, the reference points used in the HCR, additional parameters, ...)

Value

A list with 8 elements biols, fleets, covars, advice, stocks, indices, fleets.ctrl, pkgs.versions. All the elements except stocks and pkgs.versions correspond with the the updated versions of the objects used in the call to FLBEIA. stocks is a list of FLStocks object containing the perceived stocks used in the management procedure to produce the management advice. pkgs.versions is a matrix indicating the packages and package version used along the simulation.

Examples

```
## Not run:
library(FLBEIA)
library(FLAssess)          # required to use the IcesHCR. Not available for win64
library(FLash)             # required to use the IcesHCR. Not available for win64
library(ggplot2)


#-----------------------------------------------------------------
# Example with 1 stock, 1 Fleets, 1 seasons and 1 iteration: one
#-----------------------------------------------------------------
```

```
# Load the data to run FLBEIA in a one stock one fleet example using the HCR used by ICES
# in the MSY framework.
 data(one)

# The names and the class of the objects needed to run FLBEIA.
# sapply(ls(), function(x) class(get(x)))

# In this scenario a single, age-structured, stock is exploited by a single fleet with a
# unique metier.
# The fleet catches yearly exactly the adviced TAC and there is no exit-entry of vessels
# in the fishery.
# The stock abundance and exploitation level is observed without error in the observation
# model.
# There is no assessment model and the TAC advice is used through the HCR used by ICES
# in the MSY framework.



s0 <- FLBEIA(biols = oneBio,   # FLBiols object with one FLBiol element for stk1.
               SRs = oneSR,    # A list with one FLSRsim object for stk1.
               BDs = NULL,     # No Biomass Dynamic populations in this case.
            fleets = oneFl,    # FLFleets object with on fleet.
            covars = oneCv,    # covars not used
           indices = NULL,     # indices not used
            advice = oneAdv,   # A list with two elements 'TAC' and 'quota.share'
        main.ctrl = oneMainC, # A list with one element to define the start and end of
                               # the simulation.
       biols.ctrl = oneBioC,  # A list with one element to select the model to simulate
                               # the stock dynamics.
     fleets.ctrl = oneFlC,   # A list with several elements to select fleet dynamic models
                               # and store additional parameters.
      covars.ctrl = oneCvC,   # covars control not used
         obs.ctrl = oneObsC,  # A list with one element to define how the stock observed
                               # ("PerfectObs").
     assess.ctrl = oneAssC,  # A list with one element to define how the stock assessment
                               # model used ("NoAssessment").
      advice.ctrl = oneAdvC)  # A list with one element to define how the TAC advice is
                               # obtained ("IcesHCR").

# Names of the object returned by FLBEIA
names(s0)

# The default plot for FLBiol defined in FLCore
plot(s0$biols[[1]])

# Extract reference points for summaries
s0_brps <- extractBRP(oneAdvC, stkn = names(oneBio))

# Create summary data frames (biological, economic, and catch)
proj.yr     <- 2013
s0_sum      <- bioSum(s0, brp = s0_brps)
```

```
s0$fleets$fl1 <- setUnitsNA(s0$fleets$fl1) # set units to NA to avoid errors in fltSum
s0_flt      <- fltSum(s0)
s0_fltStk   <- fltStkSum(s0)


# Create several plots and save them in the working directory using 'pdf' format and
# 's0' suffix in the name.


plotFLBiols(s0$biols, pdfnm='s0', ss = 'all')
plotFLFleets(s0$fleets, pdfnm='s0', ss = 'all')
plotEco(s0, pdfnm='s0')
plotfltStkSum(s0, pdfnm='s0')

#------------------------------------------------------------
# Example with several iterations: oneIters
#------------------------------------------------------------

 # Load the same data set as before but with 3 iterations.
 # Run FLBEIA and plot the results

data(oneIt)

s1 <- FLBEIA(biols = oneItBio,   # FLBiols object with one FLBiol element for stk1.
               SRs = oneItSR,    # A list with one FLSRsim object for stk1.
               BDs = NULL,       # No Biomass Dynamic populations in this case.
            fleets = oneItFl,    # FLFleets object with on fleet.
            covars = oneItCv,    # covars not used
           indices = NULL,       # indices not used
            advice = oneItAdv,   # A list with two elements 'TAC' and 'quota.share'
        main.ctrl = oneItMainC, # A list with one element to define the start and end of
                                 # the simulation.
       biols.ctrl = oneItBioC,  # A list with one element to select the model to simulate
                                 # the stock dynamics.
       fleets.ctrl = oneItFlC,   # A list with several elements to select fleet dynamic
                                 # models and store additional parameters.
       covars.ctrl = oneItCvC,   # covars control not used
         obs.ctrl = oneItObsC,  # A list with one element to define how the stock observed
                                 # ("PerfectObs").
       assess.ctrl = oneItAssC,  # A list with one element to define how the stock
                                 # assessment model used ("NoAssessment").
      advice.ctrl = oneItAdvC)  # A list with one element to define how the TAC advice is
                                 # obtained ("IcesHCR").

# Names of the object returned by FLBEIA
names(s1)

# The default plot for FLBiol defined in FLCore
plot(s1$biols[[1]])

# Extract reference points for summaries
s1_brps <- extractBRP(oneItAdvC, stkn = names(oneItBio))
```

```
# Create summary data frames (biological, economic, and catch)
proj.yr     <- 2013
s1_bio     <- bioSum(s1, brp = s1_brps)
s1$fleets$fl1 <- setUnitsNA(s1$fleets$fl1) # set units to NA to avoid errors in fltSum
s1_flt     <- fltSum(s1)
s1_fltStk  <- fltStkSum(s1)

s1_bioQ    <- bioSumQ(s1_bio)
s1_fltQ    <- fltSumQ(s1_flt)
s1_fltStkQ <- fltStkSumQ(s1_fltStk)

s1b_bio     <- bioSum(s1, long = FALSE)
s1b_flt     <- fltSum(s1, long = FALSE)
s1b_fltStk  <- fltStkSum(s1, long = FALSE)

s1b_fltQ    <- bioSumQ(s1b_bio)
s1b_fltQ    <- fltSumQ(s1b_flt)
s1b_fltStkQ <- fltStkSumQ(s1b_fltStk)

# Create several plots and save them in the working directory using 'pdf' format and
# 's1' suffix in the name.

plotFLBiols(s1$biols, pdfnm='s1', ss = 'all')
plotFLFleets(s1$fleets, pdfnm='s1', ss = 'all')
plotEco(s1, pdfnm='s1')
plotfltStkSum(s1, pdfnm='s1')


#-------------------------------------------------------------------
# Example with 2 stock, 2 Fleets, 4 seasons and 1 iteration: multi
#-------------------------------------------------------------------

 # Load the multi data set. This dataset has 2 stocks, one stk1 is
 # age structured and the second one stk2 is aggregated in biomass.

data(multi)

 # Run FLBEIA.

s2 <- FLBEIA(biols = multiBio,   # FLBiols object with 2 FLBiol element for stk1.
               SRs = multiSR,    # A list with 1 FLSRsim object for stk1.
               BDs = multiBD,    # A list with 1 FLBDSim object for stk2.
            fleets = multiFl,    # FLFleets object with on fleet.
            covars = multiCv,    # covars not used
           indices = NULL,       # indices not used
            advice = multiAdv,   # A list with two elements 'TAC' and 'quota.share'
         main.ctrl = multiMainC, # A list with one element to define the start and end
                                 # of the simulation.
       biols.ctrl = multiBioC,  # A list with one element to select the model to simulate
                                 # the stock dynamics.
       fleets.ctrl = multiFlC,   # A list with several elements to select fleet dynamic
                                 # models and store additional parameters.
       covars.ctrl = multiCvC,   # covars control not used
```

```
          obs.ctrl = multiObsC,  # A list with one element to define how the stock observed
                                   # ("PerfectObs").
        assess.ctrl = multiAssC,  # A list with one element to define how the stock
                                   # assessment model used ("NoAssessment").
       advice.ctrl = multiAdvC)  # A list with one element to define how the TAC advice is
                                   # obtained ("IcesHCR").

# Names of the object returned by FLBEIA
names(s2)

# The default plot for FLBiol defined in FLCore
plot(s2$biols[[1]])

# Extract reference points for summaries
s2_brps <- extractBRP(multiAdvC, stkn = names(multiBio))

# Create summary data frames (biological, economic, and catch)

s2_sum      <- bioSum(s2, brp = s2_brps)
for (fl in names(s2$fleets))  # set units to NA to avoid errors in fltSum
  s2$fleets[[fl]] <- setUnitsNA(s2$fleets[[fl]])
s2_flt      <- fltSum(s2)

s2b_flt     <- fltSum(s2, byyear = FALSE)

s2_fltStk   <- fltStkSum(s2)

# Create several plots and save them in the working directory using 'pdf' format and
# 's2' suffix in the name.

plotFLBiols(s2$biols, pdfnm='s2', ss = 2)
plotFLFleets(s2$fleets, pdfnm='s2', ss = 2)
plotEco(s2, pdfnm='s2')
plotfltStkSum(s2, pdfnm='s2')

## End(Not run)
```

FLCatchesExt *Class FLCatchesExt*

Description

A list of FLCatchExt objects.

Usage

```
vFLCs(object)
```

```
## S4 method for signature 'ANY'
FLCatchesExt(object, ...)

## S4 method for signature 'missing'
FLCatchesExt(object, ...)

## S4 method for signature 'list'
FLCatchesExt(object)

is.FLCatchesExt(object, ...)

## S4 method for signature 'ANY'
is.FLCatchesExt(object, ...)

## S4 method for signature 'FLCatchesExt'
catchNames(object)

## S4 method for signature 'FLMetierExt'
catchNames(object)

## S4 method for signature 'FLMetiersExt'
catchNames(object)

## S4 method for signature 'FLFleetExt'
catchNames(object)

## S4 method for signature 'FLFleetsExt'
catchNames(object)
```

Arguments

object	An object of class FLCatchExt, list or missing.
...	Other objects to be assigned by name to the class slots.

Slots

.Data Internal S4 data representation, of class `list`.

desc As textual description of the object contents

lock Can the object be extended/trimmed? `TRUE` or `FALSE`.

names A character vector for the element names

Author(s)

The FLBEIA Team

See Also

`FLlst`, `list`, `vector`, `FLCatchExt`

FLCatchExt *FLCatchExt class and the methods to construct it.*

Description

It extends the FLCatch class defined in FLFleet package. The FLCatch class includes two extra slots alpha and beta used in the Cobb-Douglas production functions.

Usage

```
## S4 method for signature 'FLQuant'
FLCatchExt(object, range = "missing", name = "NA", desc = character(0), ...)

## S4 method for signature 'missing'
FLCatchExt(object, ...)

## S4 method for signature 'FLCatchExt,ANY,ANY,ANY'
x[i, j, k, l, m, n, ..., drop = FALSE]

## S4 replacement method for signature 'FLCatchExt,ANY,ANY,FLCatchExt'
x[i, j, k, l, m, n, ...] <- value

catchNames(object, ...) <- value

## S4 method for signature 'FLCatchExt,FLCatchExt'
addFLCatch(e1, e2)

## S4 method for signature 'FLCatchExt'
catchNames(object)

## S4 replacement method for signature 'FLCatchExt,character'
catchNames(object) <- value
```

Arguments

object, x	An object of class FLQuant, missing or FLCatchExt.
range	Numerical vector with min, max, plusgroup, minyear and maxyear elements as in FLStock object.
name	The name of the stock.
desc	The description of the object.
...	Other objects to be assigned by name to the class slots.
i, j, k, l, m, n	subindices
drop	logical. Should the dimesions be dropped?
value	Value or values to be assigned to the particular FLQuant or FLCatchExt slot.
e1, e2	FLCatchExt objects, where e2 is incorporated into e1 (see addFLCatch).

Details

The FLCatchExt object contains a representation of the catch of a fish stock as constructed for the purposes of fleet dynamic modelling. This includes information on removals (i.e. landings and discards), selectivity, weight, price and catch production parameters (catchability and elasticities).

Value

The constructors return an object of class FLCatchExt.

Slots

`landings` An FLQuant with the total landings in weight of the stock.

`landings.n` An FLQuant with the landings in numbers at age of the stock.

`landings.wt` An FLQuant with the weight at age of the landings.

`discards` An FLQuant with the total discards in weight of the stock.

`discards.n` An FLQuant with the discards in numbers at age of the stock.

`discards.wt` An FLQuant with the weight at age of the discards.

`landings.sel,discards.sel` An FLQuant with the landing/discard ogive of the metier for this stock (i.e. landings.sel corresponds to the proportion of catches landed). Elements must be between 0 and 1, with discards.sel = 1-landings.sel.

`catch.q` An FLQuant with the catchability at age of the stock for the corresponding metier. This is the catchability used in the catch production model.

`price` An FLQuant with the price at age of the stock.

`alpha` An FLQuant with the elasticity parameter at age of the stock for the corresponding metier. This is one of the parameters used in the catch production model.

`beta` An FLQuant with the elasticity parameter at age of the stock for the corresponding metier. This is one of the parameters used in the catch production model.

`name` The name of the stock.

`desc` A description of the object.

`range` The range as in other FLR objects: c("min","max","plusgroup","minyear","maxyear").

`FLFleetExt` *FLFleetExt class and the methods to construct it.*

Description

It extends the FLFleetExt class defined in FLFleet package. The only difference is that that the metiers slot is a FLMetiersExt object.

Usage

```
## S4 method for signature 'FLMetiersExt'
FLFleetExt(object, ...)

## S4 method for signature 'FLMetierExt'
FLFleetExt(object, ...)

## S4 method for signature 'FLCatchesExt'
FLFleetExt(object, ...)

## S4 method for signature 'FLCatchExt'
FLFleetExt(object, ...)

## S4 method for signature 'FLFleetExt'
FLFleetExt(object, metier, catch, ...)

## S4 method for signature 'missing'
FLFleetExt(object, ...)

## S4 method for signature 'FLFleetExt,ANY,missing,ANY'
x[i, drop = FALSE]

## S4 method for signature 'FLFleetExt,ANY,ANY,ANY'
x[i, j, drop = FALSE]

## S4 method for signature 'FLFleetExt,ANY,missing'
x[[i, drop = FALSE]]
```

Arguments

object, x	An object of class FLQuant, missing, FLFleetExt, FLCatchExt, FLCatchesExt or FLMetierExt.
...	Other objects to be assigned by name to the class slots
metier	A name of one of the elements in FLMetiersExt object.
catch	A name of one of the elements in FLCatchesExt object.
i, j	subindices.
drop	If TRUE, deletes the dimensions of an array which have only one level.

Details

The FLFleetExt object contains a representation of a fishing fleet as constructed for the purposes of fleet dynamic modelling. This includes information on effort, fixed-cost, capacity, crew-share, metiers and variable costs.

Value

The constructors return an object of class FLFleetExt.

Slots

`effort` An FLQuant with the effort of the fleet. The effort can have any units (e.g. number of fishing days, trips, hooks,...)

`fcost` An FLQuant with the fixed costs of the fleet. These costs should be given by vessel and the number of vessels by fleet must be included in the covars object.

`capacity` An FLQuant with the capacity of the fleet. Same units as in slot effort must be used.

`crewshare` An FLQuant with the crewshare of the fleet. Where crewshare is the percentage of revenues that goes to the crew.

`metiers` A FLMetiersExt with information on the fleet's metiers.

`name` The name of the stock.

`desc` A description of the object.

`range` The range as in other FLR objects: c("min","max","plusgroup","minyear","maxyear").

`FLFleetsExt` *Class FLFleetsExt*

Description

A list of `FLFleetExt` objects.

Usage

```
vFLFs(object)

## S4 method for signature 'ANY'
FLFleetsExt(object, ...)

## S4 method for signature 'missing'
FLFleetsExt(object, ...)

## S4 method for signature 'list'
FLFleetsExt(object)

is.FLFleetsExt(object, ...)

## S4 method for signature 'ANY'
is.FLFleetsExt(object, ...)
```

Arguments

`object`	An object of class FLFleetExt, list or missing.
`...`	Other objects to be assigned by name to the class slots.

Slots

.Data Internal S4 data representation, of class `list`.

desc As textual description of the object contents

lock Can the object be extended/trimmed? `TRUE` or `FALSE`.

names A character vector for the element names

Author(s)

The FLBEIA Team

See Also

`FLlst`, `list`, `vector`, `FLFleetExt`

`FLMetierExt` *FLMetierExt class and the methods to construct it.*

Description

It extends the FLMetier class defined in FLFleet package. The only difference is that that the catches slot is a FLCatchesExt object.

Usage

```
## S4 method for signature 'FLCatchExt'
FLMetierExt(catches, gear = "NA", ...)

## S4 method for signature 'FLCatchesExt'
FLMetierExt(catches, gear = "NA", ...)

## S4 method for signature 'FLQuant'
FLMetierExt(catches, gear = "NA", ...)

## S4 method for signature 'missing'
FLMetierExt(catches, gear = "NA", ...)

## S4 method for signature 'FLMetierExt,ANY,missing,ANY'
x[i, drop = FALSE]

## S4 method for signature 'FLMetierExt,ANY,missing'
x[[i, drop = FALSE]]
```

Arguments

catches, x	An object of class FLQuant, missing or FLCatchExt.
gear	A character vector with the name of the gear used in the metier.
...	Other objects to be assigned by name to the class slots
i	subindices.
drop	If TRUE, deletes the dimensions of an array which have only one level.

Details

The FLMetierExt object contains a representation of the metier of a fishing fleet as constructed for the purposes of fleet dynamic modelling. This includes information on effortshare and variable costs.

Value

The constructors return an object of class FLMetierExt.

Slots

gear A character with the gear name of a fleet's metier.

effshare An FLQuant with the effort share of a fleet's metier relative to fleet's total effort (the sum of all metiers effshares for a fleet must sum 1).

vcost An FLQuant with the varible costs of a fleet's metier. These costs should be given by vessel and based on the effort units used for the fleet's effort (within the FLFleetExt object). The number of vessels by fleet must be included in the covars object.

catches A FLCatchesExt with information on the fleet's metier catches.

name The name of the stock.

desc A description of the object.

range The range as in other FLR objects: c("min","max","plusgroup","minyear","maxyear").

FLMetiersExt *Class FLMetiersExt*

Description

A list of FLMetierExt objects.

Usage

```
vFLMs(object)

## S4 method for signature 'ANY'
FLMetiersExt(object, ...)

## S4 method for signature 'missing'
FLMetiersExt(object, ...)

## S4 method for signature 'list'
FLMetiersExt(object)

is.FLMetiersExt(object, ...)

## S4 method for signature 'ANY'
is.FLMetiersExt(object, ...)
```

Arguments

object	An object of class FLMetierExt, list or missing.
...	Other objects to be assigned by name to the class slots.

Slots

.Data Internal S4 data representation, of class `list`.

desc As textual description of the object contents

lock Can the object be extended/trimmed? `TRUE` or `FALSE`.

names A character vector for the element names

Author(s)

The FLBEIA Team

See Also

FLlst, list, vector, FLMetierExt

FLSRsim *FLSRsim class and methods*

Description

This class is used to store the necessary information to simulate the recruitment process within FLBEIA.

Arguments

...	Any of the slots in FLSRsim class.

Details

The FLSRsim class contains a representation of the recruitment process of a fish stock. This includes information on recruitment, ssb, recruitment model, uncertainty and distribution of the recruitment along seasons. The slots in this class:

Value

An object of class FLSRsim.

Slots

`rec` An FLQuant with to store the recruitment.

`ssb` An FLQuant with to store the ssb.

`covar` An FLQuants to store the covariates considered in the stock-recruitment model, one FQuant per covariate.

`uncertainty` An FLQuant with to store the uncertainty that is multiplied to the recruitment point estimate in each step of the simulation.

`proportion` An FLQuant with values between 0 and 1 to indicate how the recruitment in each of the time steps is distributed along season.

`model` A character with the name of the model to simulate the recruitment process.

`params` An array with dimensions [number of params,number of years,number of seasons,number of iteration].

`timelag` A matrix [2, number of seasons]. The element (1,j) indicates the time lag between the spawning and recruitment year and the element (2,j) the season in which the recruitment was spawn.

`name` A character with the name of the stock.

`desc` A description of the object.

`range` A numeric vector with c(min, max,plusgroup, minyear, maxyear) as in the rest of the FLR objects.

`joinIter` *Joins the iterations of several FLBEIA outputs*

Description

joinIter function allows to join different outputs of FBEIA simulations. For example when simulations are run iteration by iteration separately, with this function we can merged all the iterations into one FLBEIA output object.

Usage

```
joinIter(
  objnam,
  files,
  directory = NULL,
  Niters = 1,
  elements = "all",
  advice.ext = "TAC",
  fleets.ctrl.ext = "seasonal.share"
)
```

Arguments

objnam	Character. The name of the object that will be joined. The object must be the output of FLBEIA function.
files	A character vector with the names of the files from which objnam will be taken.
directory	The directory were the files are stored. Default value is the current directory.
Niters	A numeric vector with the number of iterations per object. If length=1, then it is assumed that all objects have the same number of iterations.
elements	The elements of the objects that must be joined. The default is to join all the objects.
advice.ext	Character. The element from advice object that will be replaced. Default is 'TAC'.
fleets.ctrl.ext	Character. The element from fleets.ctrl object that will be replaced. Default is 'seasonal.share'.

Value

A new FLBEIA output object with all the iterations joined.

Note

The files must contain a single object (named as objnam value).

---

plotbioSum *Summary plots of the FLBEIA output*

Description

Summarize the results in a plot.

Usage

```
plotbioSum(obj, stk.nam, Blim = NA, Bpa = NA, proj.yr = NA)

plotfltSum(obj, flt.nam, proj.yr = NA)
```

Arguments

obj	An object with the same format as bioSum and fltSum outputs, respectively.
stk.nam	Character with the name of the stock for which summary information will be plotted. If not defined, then the first one in obj will be selected by default.
Blim	Numeric. Blim reference point for the stock (optional argument).=NA, Bpa=NA, proj.yr=NA)
Bpa	Numeric. Bpa reference point for the stock (optional argument).
proj.yr	Numeric. Year in which projection period starts (optional argument).
flt.nam	Character with the name of the fleet for which summary information will be plotted. If not defined, then the first one in obj will be selected by default.

Details

- plotbioSum: Plot summarising information on stock. With one plot for each of the following indicators: "catch", "rec", "f" and "ssb". Input object should have the same format as the output of bioSum function.
- plotfltSum: Plot summarising information on fleet's economic indicators. With one plot for each of the following indicators: "catch", "effort", "grossValue" and "grossSurplus". Input object should have the same format as the output of fltSum function.

Value

Plot.

Examples

```
## Not run:

library(FLBEIA)

# Apply the summary plots to the examples runs in FLBEIA help page.

data(res_flbeia)
#-----------------------------------------------
# Example One: One stock, one fleet, one iter.
#-----------------------------------------------

# Biological indicators.
plotbioSum( bioSum(oneRes), Blim=800, Bpa=1200, proj.yr=2010) # bioSum output in wide format
plotbioSum( bioSum(oneRes, long = FALSE))   # bioSum output in long format
plotbioSum( bioSum(oneRes), stk.nam='stk0') # if incorrect name for the stock

# Indicators at fleet level.
plotfltSum( fltSum(oneRes), proj.yr=2010) # fltSum output in wide format
plotfltSum( fltSum(oneRes, long = FALSE)) # fltSum output in long format
plotfltSum( fltSum(oneRes), flt.nam='stk1') # if incorrect name for the fleet
plotfltSum( fltSum(oneRes, byyear = FALSE)) # although seasonal disagregation,
                                            # it is summarised by year
```

```
#-----------------------------------------------
# Example OneIters: As one but with iterations.
#-----------------------------------------------

# Biological indicators.
plotbioSum( bioSum(s1), Blim=800, Bpa=1200, proj.yr=2010) # bioSum output in wide format
plotbioSum( bioSum(s1, long = FALSE))   # bioSum output in long format
plotbioSum( bioSum(s1), stk.nam='stk0') # if incorrect name for the stock


# Indicators at fleet level.
plotfltSum( fltSum(s1), proj.yr=2010) # fltSum output in wide format
plotfltSum( fltSum(s1, long = FALSE)) # fltSum output in long format
plotfltSum( fltSum(s1), flt.nam='stk1') # if incorrect name for the fleet
plotfltSum( fltSum(s1, byyear = FALSE)) # although seasonal disagregation,
                                        # it is summarised by year

# also possible to plot information on various scenarios
sc11_bio <- bioSum(s1)
sc12_bio <- bioSum(s1, scenario='alt'); sc12_bio$value <- sc12_bio$value*1.05
plotbioSum(rbind(sc11_bio, sc12_bio), Blim=800, Bpa=1200, proj.yr=2010)

#-----------------------------------------------
# Example Multi: Two stock, two fleet, four iters.
#-----------------------------------------------

for (st in names(s2$stocks)) # one plot for each stock
  plotbioSum( bioSum(s2, scenario='s2'), stk.nam=st, proj.yr=2010)

for (fl in names(s2$fleets)) # one plot for each fleet
  plotfltSum( fltSum(s2, scenario='s2'), flt.nam=fl, proj.yr=2010)


## End(Not run)
```

`plotBRPsson` *YPR plots for a seasonal model*

Description

Function for generating YPR plots for reference points when having a seasonal model

Usage

```
plotBRPsson(obj, pdfnm = "Fbar_vs_SPR.pdf")
```

Arguments

`obj`	The output of brpsson function or a data.frame with the same structure as the runs object in the output list of brpsson function.
`pdfnm`	The name of the pdf document where plots are going to be saved.

Value

A pdf for each stock with plots.

See Also

brpsson

---

plotEco *Plots with fleets data*

Description

Return a pdf with plots using FLBEIA object (FLFleets and covars).

Usage

```
plotEco(obj, probs = c(0.95, 0.5, 0.05), pdfnm = "bc")
```

Arguments

obj	An FLBEIA object.
probs	A numeric vector with the probabilities used to calculate the quantiles.
pdfnm	The name for the pdf document will be "Eco" and pdfnm separated by a line.

Value

A pdf with capacity, costs, effort, profits by fleet.

Examples

```
## Not run:
library(FLBEIA)
library(ggplot2)

data(res_flbeia)
plotEco(oneRes, pdfnm='one')

## End(Not run)
```

`plotFLBiols` *Plots with biological data*

Description

For each stock, return a pdf with plots using FLBiols object.

Usage

```
plotFLBiols(biols, probs = c(0.95, 0.5, 0.05), pdfnm = "bc", u = 1, ss = 1)
```

Arguments

`biols`	A FLBiols object
`probs`	a numeric vector with the probabilities used to calculate the quantiles.
`pdfnm`	The name for the pdf document will be stock's name and pdfnm separated by a line.

Details

- Each pdf contains biomass in numbers at age, mean weight at age, fecundity, natural mortality, maturity, spawning, recruitment and spawning stock biomass

Value

A pdf for each stock with plots.

Examples

```
## Not run:
library(FLBEIA)
library(ggplot2)
data(res_flbeia)
plotFLBiols(oneRes$biols, pdfnm='oneRes')

## End(Not run)
```

`plotFLFleets` *Plots with fleets data*

Description

For each fleet, return a pdf with plots using FLFleets object.

Usage

```
plotFLFleets(fleets, probs = c(0.95, 0.5, 0.05), pdfnm = "bc", u = 1, ss = 1)
```

Arguments

`fleets`	A FLFleets object.
`probs`	A numeric vector with the probabilities used to calculate the quantiles.
`pdfnm`	The name for the pdf document will be the fleet's name and pdfnm separated by a line.

Details

For each fleet, the pdf contains plots of:

- Catch, discards, landings, capacity, crewshare, effort, fcost, effshare
- For each metier: landings and discards at age in numbers and mean weight, alpha, beta and catch.q

Value

A pdf for each fleet with plots.

Examples

```
## Not run:
library(FLBEIA)
library(ggplot2)
data(res_flbeia)
plotFLFleets(oneRes$fleets,pdfnm='oneFl')

## End(Not run)
```

plotfltStkSum *Plots with fltStkSum data*

Description

Return a pdf with plots with the outputs fltStkSum, usint the output of FLBEIA.

Usage

```
plotfltStkSum(obj, pdfnm)
```

Arguments

obj	FLBEIA output
pdfnm	The name for the pdf document will be 'fltStkSum-' and pdfnm.

Value

One pdf with plots on landings, discards,catch,price,quotaUpt,tacshare,discRat,quota by fleet and stock.

Examples

```
## Not run:
library(FLBEIA)
library(ggplot2)
data(res_flbeia)
plotfltStkSum(obj=oneRes,pdfnm = "oneRes")

## End(Not run)
```

revenue_flbeia *Economic summary functions.*

Description

These functions provide summary results of costs, prices and revenues. Provided data can be dessagregated by fleet or by metier depending on the selected function.

Usage

```
revenue_flbeia(fleet)

costs_flbeia(fleet, covars, flnm = NULL)

totvcost_flbeia(fleet)

totfcost_flbeia(fleet, covars, flnm = NULL)

price_flbeia(fleet, stock)
```

Arguments

fleet	An element of FLfleets object.
covars	List of FLQuants with information on covariates.
flnm	Names of the fleets.
stock	An FLStock object.

Details

- revenue_flbeia computes the revenue by fleet and metier. The revenue is computed as landings (weight) multiplied by the price.
- costs_flbeia computes total costs as the sum of fixed and variable costs.
- totvcost_flbeia computes the variable costs including crew share costs .
- totfcost_flbeia computes the total costs by vessel.
- price_flbeia computes the price by stock.

---

segremix — *Stock-Recruitment models in FLBEIA: segrexmix - Mixture of 2 segmented regression stock-recruitment models fit*

Description

Model formulation:

Usage

```
segregmix()
```

Details

$$R = \mathbf{ifelse}(S \leq b, aS, \mathbf{ifelse}(S \leq B, ab, ABuncAdd))$$

R = ifelse(S <= b, a*S, ifelse(S <= B, a*b, A*B*uncAdd)) a is the slope of the recruitment for stock levels at or below b, ab is the mean recruitment for stock levels above b and at or below B, and $ABuncAdd$ is the mean recruitment for stock levels above B. Where $a, b, A, B > 0$.

Additional stock-recruitment (SR) models to the ones provided in FLCore package.

Author(s)

The FLBEIA Team

See Also

SRModels, FLSR, FLModel

`SelAtAge` *Estimate selectivity at age*

Description

Estimates selectivity at age of an stock by a fleet and metier.

Usage

```
SelAtAge(
  biols,
  fleets,
  flnm = 1,
  mtnm = 1,
  stnm = 1,
  years,
  iter = NULL,
  restriction = 1,
  ntrials = 3
)
```

Arguments

`biols`	A FLBiols object.
`fleets`	A FLFleetsExt object.
`flnm`	A character vector with the name of the fleet for which you want to calculate selectivity.
`mtnm`	A character vector with the name of the metier for which you want to calculate selectivity.
`stnm`	A character vector with the name of the stock for which you want to calculate selectivity.
`years`	A character vector with the name of the years used to calculate selectivity.
`iter`	Numeric vector with the specific iterations to be consider. Default is taking all iterations.
`restriction`	If restriction = 1 => sum(Sa) = 1, whereas if restriction = 2 => max(Sa) = 1. Default value is 1.
`ntrials`	Numeric. If ntrials > 1, process success is checked.

Details

To calculate selectivity at age, the following formula is used:

$$C_{a,f,m} = \frac{C_{f,m}}{sum_{i=a_0,...a+} S_{i,f,m} \cdot B_i} \cdot S_{a,f,m} \cdot B_a$$

Where:

- a: age.
- f: fleet.
- m: metier.
- i: susbscript of age.
- $S_{a,f,m}$: selectivity at age 'a' for fleet 'f' and metier 'm'.
- $C_{a,f,m}$: catch (in weight) at age 'a' for fleet 'f' and metier 'm'.
- $C_{f,m}$: total catch for fleet 'f' and metier 'm'.
- B_a: biomass (in weight) at age.

Consult FLBEIA manual to see the derivation of the formula.

The equation above is nonlinear and therefore we cannot find an analytical expression for the selectivity. Rewriting the equation above for each 'a' we have de following equation:

$$sum_{i=a_0,...a+} S_{i,f,m} \cdot B_i - \frac{C_{f,m}}{C_{a,f,m}} \cdot B_a \cdot S_{a,f,m}$$

Thus we have a system of linear equations being Sa0fm,...,Sa+fm the unknown variables. The problem is that the equations in the system are not independent. Therefore, we have to remove one equation to get a system of independent equations. Furthermore, we have to add a constraint (an equation) to be able to solve the system.

To make sure that output is correct, we solve the sistem for each year and season several times (the number is set in `ntrials`), removing a different equation each time. Alway using the same constraint (`restriction=1`): $sum_{i=a_0,...a+} S_{i,f,m} = 1$ Finally, we compare the values of $S_{i,f,m}$ and we check if they are the same.

Value

A FLQuant with selectivity at age values in `years`. The rest of the years have value 0 for all ages.

setUnitsNA *Method setUnitsNA*

Description

Function to remove the units from the FLQuants of an object

Usage

```
setUnitsNA(object)

## S4 method for signature 'FLBiol'
setUnitsNA(object)

## S4 method for signature 'FLBiols'
setUnitsNA(object)

## S4 method for signature 'FLFleetExt'
setUnitsNA(object)

## S4 method for signature 'FLFleetsExt'
setUnitsNA(object)
```

Arguments

object	An object of class FLBiol, FLBiols, FLFleetExt or FLFleetsExt object.

Value

The same object with the units equal to NA in all the FLQuant slots.

Examples

```
data(one)

st <- setUnitsNA(oneBio)
fl <- setUnitsNA(oneFl)

data(multi)

biols <- setUnitsNA(multiBio)
fleets <- setUnitsNA(multiFl)

data(res_flbeia)

res_biols  <- setUnitsNA(multiRes$biols)
res_fleets <- setUnitsNA(multiRes$fleets)
```

stock.fleetInfo *stock.fleetInfo*

Description

Indicates which stocks are caugth by each fleet-metier combination.

Usage

```
stock.fleetInfo(fleets)
```

Arguments

fleets	is an objetc of class FLFleetsExt.

Value

Return a matrix with rownames equal to the stocks names and colnames equal to names of fleet and metier. If element (i,j) is equal to 0, the stock (i) is not caught by fleet/metier (j).

Examples

```
## Not run:
data(multi)
stock.fleetInfo(fl1)

## End(Not run)
```

tlandStock *Extract landings, discard or catch at age*

Description

Extract landings, discards or catch at age of a stock from a FLFleetExt or FLFleetsExt object.

Usage

```
tlandStock(obj, stknm)

tdiscStock(obj, stknm)

landStock.f(obj, stock)

discStock.f(obj, stock)

catchStock.f(obj, stock)
```

```
landWStock.f(obj, stock)

discWStock.f(obj, stock)

catchWStock.f(obj, stock)

landStock(obj, stock)

discStock(obj, stock)

catchStock(obj, stock)

landWStock(obj, stock)

discWStock(obj, stock)

catchWStock(obj, stock)
```

Arguments

obj	a FLFleetExt (.f) or FLFleetsExt object
stknm	stock name
stock	stock name

Details

- landStock.f: Extract total landings at age in numbers of a stock from a FLFleetExt obj.
- discStock.f: Extract total discards at age in numbers of a stock from a FLFleetExt obj.
- catchStock.f: Extract total catch at age in numbers of a stock from a FLFleetExt obj.
- landWStock.f: Extract total landings at age in weight of a stock from a FLFleetExt obj.
- discWStock.f: Extract total discards at age in weight of a stock from a FLFleetExt obj.
- catchWStock.f: Extract total catch at age in weight of a stock from a FLFleetExt obj.
- landStock: Extract total landings at age in numbers of a stock from a FLFleetExts obj.
- discStock: Extract total discards at age in numbers of a stock from a FLFleetExts obj.
- catchStock: Extract total catch at age in numbers of a stock from a FLFleetExts obj.
- landWStock: Extract total landings at age in weight of a stock from a FLFleetExts obj.
- discWStock: Extract total discards at age in weight of a stock from a FLFleetExts obj.
- catchWStock: Extract total catch at age in weight of a stock from a FLFleetExts obj.
- tlandStock: Total landings of a stock across fleets and metiers
- tdiscStock: Total discards of a stock across fleets and metiers

Value

A FLQuant object with landings, discards or catch (total or at age).

Examples

```
## Not run:
library(FLBEIA)
data(one)
landWStock.f(oneFl$fl1,"stk1")
landWStock(oneFl,"stk1")

## End(Not run)
```

`unit2age` *Translates unit dimension in an FQuant into age dimension*

Description

This function transforms a FQuant with several 'unit' dimension into an array an unique 'unit' dimension and no 'area' dimension. Moving 'unit' to 'age' dimension. This is usefull when we use the 'unit' dimension to store the different seasonal cohorts.

Usage

```
unit2age(Q)
```

Arguments

`Q`	A FLQuant. The object must be the output of FLBEIA function.

Value

A 4-dimensional array with length c(d[1]*d[3], d[2], d[4], d[6]), where d is the dimension of the FLQuant.

Note

The files must contain a single object (named as `objnam` value).

`updateFLBiols` *Update the FLBiols object to the FLCore 2.6*

Description

Updates an old FLBiols (where slots fec and mat from each FLBiol are FLQuants), into the new version of the class, where slots fec and mat are of class `predictModel`.

Usage

```
updateFLBiols(biols)
```

Arguments

biols	A FLBiols object.

Value

A new FLBEIA output object with all the iterations joined.

See Also

predictModel

---

wtadStock *Mean weight at age in discards for a stock across fleets and metiers*

Description

Mean weight at age in discards for a stock across fleets and metiers

Usage

```
wtadStock(obj, stknm)
```

Arguments

obj	An object of class FlFleetsExt.
stknm	Character. The name of the stock for which we want to calculate mean weight-at-age.

Value

A FLQuant with mean weight-at-age values.

See Also

wtalStock

`wtalStock` *Mean weight at age in landings for a stock across fleets and metiers*

Description

Mean weight at age in landings for a stock across fleets and metiers

Usage

```
wtalStock(obj, stknm)
```

Arguments

`obj`	An object of class FlFleetsExt.
`stknm`	Character. The name of the stock for which we want to calculate mean weight-at-age.

Value

A FLQuant with mean weight-at-age values.

See Also

`wtadStock`

Index