Package 'FLBEIA'

February 1, 2019

Title Bio-Economic Impact Assessment of Management Strategies using FLR

Version 1.15.2

Date 2017-10-02

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.

Depends methods, R(>= 3.2.3), FLCore (>= 2.6.5), FLFleet, ggplot2

Suggests R.rsp, FLXSA, FLa4a, Matrix, spict, FLash, FLAssess, nloptr, triangle, mvtnorm

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

License GPL-2

VignetteBuilder R.rsp

RoxygenNote 6.0.1

NeedsCompilation no

Author Dorleta Garcia [aut, cre]

Maintainer Dorleta Garcia <dgarcia@azti.es>

R topics documented:

`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, ...)

aneHCRE(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, ...)

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

ghlHCR(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 and pidHCRItarg.
- Model-Based HCRs: aneHCRE, annualTAC, CFPMSYHCR, F2CatchHCR, FroeseHCR, IcesHCR, MAPHCR, neaMAC_ltmp.

- aneHCRE: The HCR used in the bay of biscay anchovy long term management plan.
- 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 2010.
- 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.

- 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

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

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

## 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 = TRUE, scenario = "bc")

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

fltSum(obj, flnms = "all", years = dimnames(obj$biols[[1]]@n)$year,
```

```
  byyear = TRUE, long = TRUE, 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 = TRUE,
  scenario = "bc")

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 = TRUE,
  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 = TRUE,
  scenario = "bc")

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

advSum(obj, stknms = catchNames(obj$fleets),
  years = dimnames(obj$biols[[1]]@n)[[2]], long = TRUE, 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 = TRUE, 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 = TRUE, 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 TRUE.
scenario	a character string with the name of the scenario corresponding with obj. Default bc.
prob	a numeric vector with the probabilities used to calculate the quantiles.
flnms	Names of the fleet for which the indicators will be calculated.
byyear	logical. The indicators should be provided at season or year level? Default TRUE.
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.
Prflim	named numeric vector with one element per fleet in flnms. The limit 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 fleet. 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

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.
#-----------------------------------------------
oneRes_bio    <- bioSum(oneRes)
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)

# Wide format
oneRes_bio    <- bioSum(oneRes, long = FALSE, years = ac(2016:2020))
```

```
oneRes_flt    <- fltSum(oneRes, long = FALSE, years = ac(2016:2020))
oneRes_fltStk <- fltStkSum(oneRes, long = FALSE, years = ac(2016:2020))
oneRes_mt     <- mtSum(oneRes, long = FALSE, years = ac(2016:2020))
oneRes_mtStk  <- mtStkSum(oneRes, long = FALSE, years = ac(2016:2020))
oneRes_adv    <- advSum(oneRes, long = FALSE, 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. No seasonal disagregation available for bio and adv summaries.

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

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


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

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

#------------------------------------------------
# Example Multi: Two stock, two fleet, four iters.
#------------------------------------------------
multiRes_bio    <- bioSum(multiRes)
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)

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


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


## End(Not run)
```

`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.
`first.year`	The first year in which advice is calculated.
`last.year`	The last year in which advice is calculated.

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, name = NA, ages, hist.yrs, sim.yrs,
  fbar = NULL, mean.yrs, excel = TRUE, 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.
`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.
`excel`	Logical. `TRUE` (default), if the data is provided in an Excel file and `FALSE`, if an RData object is used instead.
`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.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 dimensioan 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.

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)
```

`F_flbeia` *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.

Usage

```
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

`obj`	The output of the FLBEIA function.
`years`	The years for which the indicators are extracted.

Details

- B_flbeia this function computes SSB.
- F_flbeia ithis 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 ithis function computes the discards by fleets 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.

`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 4production 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 Estrategy 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 FLSRSim 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 addit
       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 ("NoA
      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]])

# Create summary data frames (biological, economic, and catch)
proj.yr     <- 2013
s0_sum      <- bioSum(s0)
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$biols, pdfnm='s0')
plotFLFleets(s0$fleets, pdfnm='s0')
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 add
       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 ("N
      advice.ctrl = oneItAdvC)      # A list with one element to define how the TAC advice is obtained ("IcesH

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

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

```
# Create summary data frames (biological, economic, and catch)
proj.yr     <- 2013
s1_bio     <- bioSum(s1)
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$biols, pdfnm='s1')
plotFLFleets(s1$fleets, pdfnm='s1')
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 add
     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 ("N
    advice.ctrl = multiAdvC)      # A list with one element to define how the TAC advice is obtained ("IcesH

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

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

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

s2_sum      <- bioSum(s2)
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$biols, pdfnm='s2')
plotFLFleets(s2$fleets, pdfnm='s2')
plotEco(s2, pdfnm='s2')
plotfltStkSum(s2, pdfnm='s2')

## End(Not run)
```

FLCatchesExt-class *FLCatchExt and FLCatchesExt classes and the methods to construct it.*

Description

They extend the FLCatch and FLCatches classes defined in FLFleets package. the FLCatch class includes two extra slots alpha and beta used in the Cobb-Douglas production functions.

Usage

```
## 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 '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

## 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.
...	Other objects to be assigned by name to the class slots
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.
i, j, k, l, m, n	subindices
drop	logical. Should the dimesions be dropped?

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 of dimension [1,num. years, num. units, numb.season, 1, numb. iters] with the total landings in weight of the stock.

landings.n An FLQuant of dimension [num. ages ,num. years, num. units, numb.season, 1, numb. iters] with the landings in numbers at age of the stock.

landings.wt An FLQuant of dimension [num. ages ,num. years, num. units, numb.season, 1, numb. iters] with the weight at age of the landings.

landings.sel An FLQuant of dimension [num. ages ,num. years, num. units, numb.season, 1, numb. iters] with the retention ogive of the metier for this stock. Elements must be between 0 and 1. landings.sel = 1-discards.sel.

`discards` An FLQuant of dimension [1,num. years, num. units, numb.season, 1, numb. iters] with the total discards in weight of the stock.

`discards.n` An FLQuant of dimension [num. ages ,num. years, num. units, numb.season, 1, numb. iters] with the discards in numbers at age of the stock.

`discards.wt` An FLQuant of dimension [num. ages ,num. years, num. units, numb.season, 1, numb. iters] with the weight at age of the discards

`discards.sel` landings.sel An FLQuant of dimension [num. ages ,num. years, num. units, numb.season, 1, numb. iters] with the discard ogive of the metier for this stock. Elements must be between 0 and 1. discards.sel. = 1-landings.sel.

`catch.q` An FLQuant of dimension [num. ages ,num. years, num. units, numb.season, 1, numb. iters] 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 of dimension [num. ages ,num. years, num. units, numb.season, 1, numb. iters] with the price at age of the stock.

`alpha` An FLQuant of dimension [num. ages ,num. years, num. units, numb.season, 1, numb. iters] with the elsaticity 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 of dimension [num. ages ,num. years, num. units, numb.season, 1, numb. iters] with the elsaticity 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")

`FLFleetsExt-class` *FLFleetExt and FLFleetsExt classes and the methods to construct it.*

Description

They extend the FLFleetExt and FLFleetsExt classes defined in FLFleets package. The only different is that that the metiers slot is a FLMetiersExt object

Usage

```
## 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, ...)
```

```
## 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	An object of class FLQuant, missing or FLMetierExt.
...	Other objects to be assigned by name to the class slots
metier	A name of one of the elements in FLMetiers object.
catch	A name of one of the elements in FLCatches object.
range	Numerical vector with min, max, plusgroup, minyear and maxyear elements as in FLStock object.
name	The name of the fleet.
desc	The description of the object.
metiers	An object of class FLMetierExt or FLMetiersExt.

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.

FLMetiersExt-class *FLMetierExt and FLMetiersExt classes and the methods to construct it.*

Description

They extend the FLMetier and FLMetiers classes defined in FLFleets package. The only different is that that the catches slot is a FLCatchesExt object

Usage

```
## 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, ...)

## 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

object	An object of class FLQuant, missing or FLCatchExt.
...	Other objects to be assigned by name to the class slots
catches	An object of class FLCatchesExt.
gear	A character vector with the name of the gear used in the metier.

range	Numerical vector with min, max, plusgroup, minyear and maxyear elements as in FLStock object.
name	The name of the metier.
desc	The description of the object.

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.

FLSRsim *FLSRsim class*

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.

`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)
```

plotEco *Plots with fleets data*

Description

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

Usage

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

Arguments

obj	An FLBEIA object.
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, prob = c(0.95, 0.5, 0.05), pdfnm = "bc")
```

Arguments

biols	A FLBiols object
prob	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, pdfname='oneRes')

## End(Not run)
```

---

plotFLFleets *Plots with fleets data*

Description

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

Usage

```
plotFLFleets(fleets, prob = c(0.95, 0.5, 0.05), pdfnm = "bc")
```

Arguments

fleets	A FLFleets object.
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(oneFl,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	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.
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.

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 of a stock from a FLFleetExt obj.
- discStock.f: Extract total discards at age of a stock from a FLFleetExt obj.
- catchStock.f: Extract total catch at age 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 of a stock from a FLFleetExts obj.
- discStock: Extract total discards at age of a stock from a FLFleetExts obj.
- catchStock: Extract total catch at age 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 discard 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)
```

Index