Conditioning FLBEIA with smart conditioning functions

FLBEIA team

AIM

FLBEIA provides a battery of tutorials for learning how to use this software. This tutorial is a practical guide about how to condition FLBEIA using the smart conditioning functions. The smart conditioning functions facilitate, save time and help to avoid possible errors later on when FLBEIA is ran. The tutorial will start showing how to condition the example `one` a simple example with single stock, single fleet and one iteration. FLBEIA library also includes some plots in order to show a fast way to verify if the data are introduced correctly.

Afterwards, a more complicated example will be shown with 2 stocks, 2 fleets, 4 seasons and 3 iterations. The objects created here are the same as in the example `multi`. The examples `one` and `multi` are part of the examples used in other tutorials.

Required packages to run this tutorial

To follow this tutorial you should have installed the following packages

- CRAN: ggplot2
- FLR: FLCore, FLBEIA, FLFleet, ggplotFL

If you are using Windows, please use 32-bit R version because some of the packages do not work in 64-bit.

```
install.packages(c("ggplot2"))
install.packages(c("FLCore", "FLFleets", "FLash", "FLAssess", "FLXSA", "FLBEIA",
    "ggplotFL"), repos = "http://flr-project.org/R")
```

Load all necessary packages.

```
library(FLCore)
```

```
## Loading required package: MASS
```

```
## Loading required package: lattice
```

```
## FLCore (Version 2.6.5, packaged: 2017-09-18 09:08:34 UTC)
```

```
library(FLash)
```

```
##
## Attaching package: 'FLash'
```

```
## The following object is masked from 'package:FLCore':
##
##     fwd
```

```
library(FLAssess)
library(FLFleet)
library(FLXSA)
library(FLBEIA)
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'ggplot2'
## The following object is masked from 'package:FLCore':
##
##     %+%
```

```
library(ggplot2)
```

Conditioning example one

Description of the example one

This example is one of the simplest possible implementation of **FLBEIA**. The Operating Model (OM) is formed by a single age-structured stock. Only one fleet which activity is performed in an unique metier and the time step is annual. The historic data go from 1990 to 2009 and projection period from 2010 to 2025.

- Operating model:
  - Population dynamics: Age structured population growth
  - SR model: Beverthon and Holt autoregressive/segmented regression
- Management Procedure:
  - Perfect observation
  - No assessment
  - Ices HCR

Input data

Download the input data from ConditioningOne.zip. The data are in '.csv' format and the example shows how to transform the data in the format required to run FLBEIA.

Conditioning

`FLBEIA` requires a number of input arguments to run a simulation; such as `biols`, `SRs`, `BDs`, `fleets`, `covars`, `indices`,`advice`, `main.ctrl`, `biols.ctrl`, `fleets.ctrl`, `covars.ctrl`, `obs.ctrl`, `assess.ctrl` and `advice.ctrl`.

These objects contain biological and economical historical and projection data, and also point the functions that are going to be used by the model. Here we introduce and explain the smart conditioning functions that have been generated to facilitate the creation of these objects.

When the data in the example `one` are loaded, with the `ls()` command we can see the objects stored in `one`, which are those needed to call to run `FLBEIA`, and these are some of the objects that will be created in this tutorial.

```
data(one)
ls()
```

The smart conditioning functions are listed and described in the Manual of **FLBEIA**, within the 'doc' folder of the package installation or typing `help(package = FLBEIA)` in the R console. The description of the smart conditioning functions are listed below.

- `create.biols.data`: It generates a `FLBiol` object for each stock, and includes all of them in a `FLBiols` object. It returns an object that could be used as `biols` argument in `FLBEIA` function. The function filled historical data of weight, abundance, natural mortality, fecundity, spawning, maturity and recruitment.

In the projection years, weight, natural mortality, fecundity, maturity and spawning, are assumed equal to the average of the historical years that the user specifies in `stk_biol.proj.avg.yrs`.

- `create.SRs.data`: It generates a list with `FLSRsim` objects and returns an object that could be used as `SRs` argument in `FLBEIA` function. This function does not calculate stock-recruitment function's parameter values; therefore, they must be calculated previously by the user. In case that the proportion of spawning per season is not defined in the projection years, then it is assumed equal to the average of the historical years range that the user defines. If uncertainty is not an input, then there is not uncertainty.
- `create.BDs.data`: It generates a list with `FLBDsim` objects. It returns an object that could be used as `BDs` argument in `FLBEIA` function. This function does not calculate biomass dynamics function's parameter values, so they must be introduced by the user. If uncertainty is not an input, then the function assumes no uncertainty.
- `create.fleets.data`: It generates an `FLFleet` object for each fleet and includes all of them in an `FLFleets` object. It returns an object that could be used as `fleets` argument in `FLBEIA` function. The function requires historical data of effort per fleet, effort share between m?tiers, landings and weight at age per stock. Notice that when the input data of landings weight are not especified then it takes the weight values specified to the stock in `create.biols.data`. In case that discards data are available, then its weight is assumed the same as for landings. The function assumes that when historical data of effort, fixed cost, capacity or crewshare are introduced, then the projection values of each of them are the average of the years that the user sets in `fl.proj.avg.yrs`; in the case of effort share and variable cost per metier, is set in `fl.met_proj.avg.yrs`; and in the case of landings at age, discards at age and price, in `fl.met.stk_proj.avg.yrs`. If the Cobb-Douglas parameters, `alpha`, `beta` and `q`, are not introduced as inputs, then they are created assumming that `alpha` and `beta` are equal to 1 and `q` is the ratio between total catch and effort per metier multiplied by the stock abundance.
- `create.indices.data`: It generates a list with all the stocks and for each stock a list with `FLIndex` or `FLIndexBiomass` objects. It returns an object that could be used as `indices` argument in `FLBEIA` function.
- `create.advice.data`: It generates a list with the advice for each of the stocks. It returns an object that could be used as `advice` argument in `FLBEIA` function. In case that the values of TAC and TAE are not introduced in the projection years, then the model assumes that they are equal to the average of the historical data that the user defines in `stk_advice.avg.yrs`, although not always is necessary to have as input the TAC value of the projection. When quota share is not defined in the projection years, then the function calculates it for each stock as the ratio between catch per fleet and total catch.
- `create.biols.ctrl`: It creates an object with the name of the growth function for each stock. The object that returns this function can be used as `biols.ctrl` argument in `FLBEIA` function.
- `create.fleets.ctrl`: It creates an object with the fleet dynamics function that is applied for each fleet. The object that returns this function can be used as `fleets.ctrl` argument in `FLBEIA` function.
- `create.covars.ctrl`: It creates an object with the function that is applied to the covariate. The object that returns this function can be used as `covars.ctrl` argument in `FLBEIA` function.
- `create.obs.ctrl`: It creates a function with the observed function for each stock. The object that returns this function can be used as `obs.ctrl` argument in `FLBEIA` function.
- `create.advice.ctrl`: It creates an object with the harvest control rule for each stock, and its parameter values. The object that returns this function can be used as `advice.ctrl` argument in `FLBEIA` function.
- `create.assess.ctrl`: It creates an object with the name of the kind of assessment that is applied to each stock. The object that returns this function can be used as `assess.ctrl` argument in `FLBEIA` function.

biols:

An FLBiols object that contains biological information of stk1 relative to 'real' biological population within the OM.

```
# Set Simulation parameters related with time

first.yr <- 1990
proj.yr <- 2010
last.yr <- 2025
yrs <- c(first.yr = first.yr, proj.yr = proj.yr, last.yr = last.yr)

# Set names, age, dimensions

fls <- c("fl1")

stks <- c("stk1")

fl1.mets <- c("met1")
fl1.met1.stks <- c("stk1")

# all stocks the same

ni <- 1
it <- 1:ni
ns <- 1

# stock stk1
stk1.age.min <- 1
stk1.age.max <- 12
stk1.unit <- 1

# Data: stk1_n.flq, m, spwn, fec, wt

# stock stk1
stk1_n.flq <- iter(as.FLQuant(read.csv(file = "data/stk1_n.csv")), it)
stk1_m.flq <- iter(as.FLQuant(read.csv(file = "data/stk1_m.csv")), it)
stk1_spwn.flq <- iter(as.FLQuant(read.csv(file = "data/stk1_spwn.csv")), it)
stk1_fec.flq <- iter(as.FLQuant(read.csv(file = "data/stk1_fec.csv")), it)
stk1_wt.flq <- iter(as.FLQuant(read.csv(file = "data/stk1_wt.csv")), it)
stk1_mat.flq <- stk1_fec.flq
stk1_mat.flq[] <- 1

stk1_range.min <- 1
stk1_range.max <- 12
stk1_range.plusgroup <- 12
stk1_range.minyear <- 1990
stk1_range.minfbar <- 4
stk1_range.maxfbar <- 9

# Projection biols: weight,fecundity,mortality and spawning

stk1_biol.proj.avg.yrs <- c(2007:2009)
```

```
# Create the object

stks.data <- list(stk1 = ls(pattern = "^stk1"))

biols <- create.biols.data(yrs, ns, ni, stks.data)
plotFLBiols(biols, pdfnm = "s0")
```

SRs:

The stock recruitment relationship, not needed in the case of population aggregated in biomass.The model is `bevholtAR1`, a regression stock recruitment model with autoregressive normal log residuals of first order. There are several SR models available in FLBEIA: `geomean`, `bevholt`, `ricker`, `segreg`, `shepherd`, `rickerAR1`, `segregAR1`, `cushing`, `bevholtSV`, `rickerSV`, `segregSV`, `shepherdSV`, `cushingSV`, `rickerCa`, `hockstick`, `redfishRecModel` and `ctRec`.

```
stk1_sr.model <- "bevholtAR1"
stk1_params.n <- 3
stk1_params.array <- xtabs2(data ~ param + year + season + iter, data = read.csv(file = "data/stk1_para
    exclude = NULL, na.action = na.pass)[, , , it, drop = F]
stk1_params.name <- c("a", "b", "c")
stk1_rec.flq <- iter(as.FLQuant(read.csv(file = "data/stk1_rec.csv")), it)
stk1_ssb.flq <- iter(as.FLQuant(read.csv(file = "data/stk1_ssb.csv")), it)
stk1_uncertainty.flq <- iter(as.FLQuant(read.csv(file = "data/stk1_uncertainty.csv")),
    it)
stk1_proportion.flq <- iter(as.FLQuant(read.csv(file = "data/stk1_proportion.csv")),
    it)
stk1_prop.avg.yrs <- ac(2006:2008)
stk1_timelag.matrix <- matrix(c(1, 1), nrow = 2, ncol = 1, dimnames = list(c("year",
    "season"), "all"))

# FLBEIA input object: SRs

stks.data <- list(stk1 = ls(pattern = "^stk1"))

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

fleets:

It is a FLFleetExt object that contains information relative to the fleet fl1 (effort, costs, capacity...) and information relative to the metier met1 (effort share, varaible costs, parameters of the Cobb Dougras function).

```
# Data per fleet effort, crewshare, fcost, capacity Data per fleet and
# metier effshare, vcost Data per fleet, metier and stock landings.n,
# discards.n,landings.wt, discards.wt, landings, discards, landings.sel,
# discards.sel, price

fl1_effort.flq <- iter(as.FLQuant(read.csv(file = "data/fl1_effort.csv")), it)
fl1_capacity.flq <- iter(as.FLQuant(read.csv(file = "data/fl1_capacity.csv")),
    it)
fl1_fcost.flq <- iter(as.FLQuant(read.csv(file = "data/fl1_fcost.csv")), it)
fl1_crewshare.flq <- iter(as.FLQuant(read.csv(file = "data/fl1_crewshare.csv")),
    it)
```

```
fl1.met1_effshare.flq <- iter(as.FLQuant(read.csv(file = "data/fl1.met1_effshare.csv")),
    it)

fl1.met1.stk1_landings.n.flq <- iter(as.FLQuant(read.csv(file = "data/fl1.met1.stk1_landings.n.csv")),
    it)
fl1.met1.stk1_discards.n.flq <- iter(as.FLQuant(read.csv(file = "data/fl1.met1.stk1_discards.n.csv")),
    it)
# fl1.met1.stk1_alpha.flq <- iter(as.FLQuant(read.csv(file =
# 'data/fl1.met1.stk1_alpha.csv')) ,it) fl1.met1.stk1_beta.flq <-
# iter(as.FLQuant(read.csv(file = 'data/fl1.met1.stk1_beta.csv')) ,it)
# fl1.met1.stk1_catch.q.flq <- iter(as.FLQuant(read.csv(file =
# 'data/fl1.met1.stk1_catch.q.csv')) ,it)

# Projection fleets: fl1
fl1_proj.avg.yrs <- c(2008:2009)
fl1.met1_proj.avg.yrs <- c(2008:2009)
fl1.met1.stk1_proj.avg.yrs <- c(2006:2008)

# create fleets object

fls.data <- list(fl1 = ls(pattern = "^fl1"))

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

plotFLFleets(fleets, pdfnm = "s0")
```

advice:

List containing information on management advice; total allowable catch 'TAC' and quota share.

```
# advice:TAC/TAE/quota.share

stk1_advice.TAC.flq <- iter(as.FLQuant(read.csv(file = "data/stk1_advice.tac.csv")),
    it)
stk1_advice.quota.share.flq <- iter(as.FLQuant(read.csv(file = "data/stk1_advice.quota.share.csv")),
    it)
stk1_advice.avg.yrs <- c(2006:2008)

# create advice object
stks.data <- list(stk1 = ls(pattern = "^stk1"))
advice <- create.advice.data(yrs, ns, ni, stks.data, fleets)
```

indices:

There is no index for this example.

```
indices <- NULL
```

main.ctrl:

Controls the settings with the initial and final year of the proyected period.

```
main.ctrl <- list()
main.ctrl$sim.years <- c(initial = proj.yr, final = last.yr)
```

biols.ctrl:

A list that controls the biological operation model. The growth function of stk1 is defined as Age Structured Population Growth `ASPG`. The function `ASPG` describes the evolution of an age structured population using an exponential survival equation for existing age classes and a stock-recruitment relationship to generate the recruitment. In FLBEIA there are other models available: `fixedPopulation` and `BDPG`.

```
growth.model <- c("ASPG")
biols.ctrl <- create.biols.ctrl(stksnames = stks, growth.model = growth.model)
```

fleets.ctrl:

The basic structure of the fleets.ctrl object; the effort dynamics is Simple Mixed Fisheries Behaviour `SMFB`. `SMFB` is a simplified version of the behavior of fleets that work in a mixed fisheries framework. The function is seasonal and assumes that effort share among metiers is given as input parameter. There are also others effort model defined in FLBEIA [`fixedEffort`, `SSFB`, `MaxProfit`, `MaxProfitSeq`]. Summarizing, the fleet catches yearly exactly the adviced TAC and there is no exit-entry of vessels in the fishery.

```
n.fls.stks <- 1
fls.stksnames <- "stk1"
effort.models <- "SMFB"
effort.restr.fl1 <- "stk1"
restriction.fl1 <- "catch"
catch.models <- "CobbDouglasAge"
capital.models <- "fixedCapital"
flq.stk1 <- FLQuant(dimnames = list(age = "all", year = first.yr:last.yr, unit = stk1.unit,
    season = 1:ns, iter = 1:ni))
fleets.ctrl <- create.fleets.ctrl(fls = fls, n.fls.stks = n.fls.stks, fls.stksnames = fls.stksnames,
    effort.models = effort.models, catch.models = catch.models, capital.models = capital.models,
    flq = flq.stk1, effort.restr.fl1 = effort.restr.fl1, restriction.fl1 = restriction.fl1)

fleets.ctrl$fl1$stk1$discard.TAC.OS <- FALSE
fleets.ctrl$fl1$restriction <- "landings"
```

advice.ctrl:

Controls the behaviour of advice management procedure.It is a list with one element per stock and one more element for each fleet. The selected model is `IcesHCR`.The function represents the HCR used by ICES to generate TAC advice in the MSY framework. It is a biomass based HCR, where the TAC advice depends on F in relation to several reference points.

```
HCR.models <- c("IcesHCR")
ref.pts.stk1 <- matrix(rep(c(548.6296271, 768.0814779, 0.1057783), 3), 3, ni,
    dimnames = list(c("Blim", "Btrigger", "Fmsy"), 1:ni))
advice.ctrl <- create.advice.ctrl(stksnames = stks, HCR.models = HCR.models,
    ref.pts.stk1 = ref.pts.stk1, first.yr = first.yr, last.yr = last.yr)

advice.ctrl[["stk1"]][["sr"]] <- list()
advice.ctrl[["stk1"]][["sr"]][["model"]] <- "geomean"
```

```
advice.ctrl[["stk1"]][["sr"]][["years"]] <- c(y.rm = 2, num.years = 10)
advice.ctrl$stk1$AdvCatch <- rep(TRUE, length(first.yr:last.yr))  #TRUE advice in catches, FALSE advice
names(advice.ctrl$stk1$AdvCatch) <- as.character((first.yr:last.yr))
```

assess.ctrl:

Defines the name of the assessment model to be used for each stock. In the current example there is no asssessment `NoAssessment`.

```
assess.models <- "NoAssessment"
assess.ctrl <- create.assess.ctrl(stksnames = stks, assess.models = assess.models)
assess.ctrl[["stk1"]]$work_w_Iter <- TRUE
```

obs.ctrl:

There is no assessment model in this examples,therefore, the management advice is just based on the 'observed' population. A `perfecObs` model is defined. This function does not introduce any observation uncertainty in the observation of the different quantities stored in the `FLStock` or `FLFLeetsExt` objects.

```
stkObs.models <- "perfectObs"
flq.stk1 <- FLQuant(dimnames = list(age = "all", year = first.yr:last.yr, unit = stk1.unit,
    season = 1:ns, iter = 1:ni))

obs.ctrl <- create.obs.ctrl(stksnames = stks, stkObs.models = stkObs.models,
    flq.stk1 = flq.stk1)
```

BDs/covars/covars.ctrl/ NULL objects

```
covars.ctrl <- NULL
BDs <- NULL
covars <- NULL
```

FLBEIA input objects

```
save(biols, SRs, BDs, fleets, covars, indices, advice, main.ctrl, biols.ctrl,
    fleets.ctrl, covars.ctrl, obs.ctrl, assess.ctrl, advice.ctrl, file = "input_S0_1it.RData")
```

Run FLBEIA

```
s0 <- FLBEIA(biols = biols, SRs = SRs, BDs = BDs, fleets = fleets, covars = covars,
    indices = indices, advice = advice, main.ctrl = main.ctrl, biols.ctrl = biols.ctrl,
    fleets.ctrl = fleets.ctrl, covars.ctrl = covars.ctrl, obs.ctrl = obs.ctrl,
    assess.ctrl = assess.ctrl, advice.ctrl = advice.ctrl)
```

```
## Note: method with signature 'FLQuant#ANY#ANY' chosen for function 'ifelse',
##  target signature 'FLQuant#numeric#FLQuant'.
##  "ANY#ANY#FLQuant" would also be valid
```

Results FLBEIA

```
names(s0)
plotFLBiols(s0$biols, pdfnm = "s0_res")
plotFLFleets(s0$fleets, pdfnm = "s0_res")
```

EXERCISES

Create the objects to condition the example **multi**.

Description of the example multi

This test case analyzes the dynamics of two fictitious stocks (stk1 and stk2), with two fleets (fl1 and fl2). Each of the fleets has two metiers (met1 and met2) and all the metiers catch both stocks. This case study simulates the management using an ICES harvest control rule for stock stk1 and annual TAC for stk2. The effort function of both fleets is different; fixed effort is assumed for fleet fl1 and simple mixed fisheries behavior, limited by the minimum catch of both stocks, for fleet fl2. There is no assessment in this study and perfect observation is assumed.

The biological historical data available for both stocks range from 1990 to 2008. The simulation time covers from 1990 to 2025, with 2009 as the first projection year. Biological data are described by stock abundance, weight, spawning, fecundity and natural mortality. Biological data of stock stk1 are age structured in a range from 0 to 15 years, but not for stock stk2, which is modelled in biomass. The minimum age to calculate the average f for stock stk1 is set at age 1 and the maximum at age 5. Stock stk1 has 4 spawning seasons, while stock stk2 only one, then we set different units for both. The values of weight, spawning, fecundity and natural mortality in the projection period are assumed to be equal to the average between 2006 and 2008 for both stocks.

Both fleets have historical information on effort and capacity, but only fleet fl2 has fixed cost data. For each fleet and metier, effort share data are available and only for fleet fl2 are variable costs. For each fleet and metier, there is age structured data for stock stk1 and in biomass for stk2. Both fleets have landings and discards data for each stock, but in the case of stock stk2 discards data of metier met2 in fleet fl1 are missing. Cobb-Douglas parameters (alpha,beta,q) are not introduced as input; therefore the function create.fleets.data calls calculate.CBparam function to calculate them.

Since the biological and economical historic data of stock stk1 are age specific, then the model allows using age-structured population growth and catch model, while in the case of stock stk2 they must be based on biomass. Beverton-Holt model is applied as stock-recruitment relationship for stock stk1, with 4 spawning seasons, and Pella-Tomlinson biomass dynamics model for stock stk2. In both cases the parameters in the projection period are introduced as input.

Input data

Download the input data from ConditioningMulti.zip.

More information

- You can submit bug reports, questions or suggestions on this tutorial at https://github.com/flr/doc/issues.
- Or send a pull request to https://github.com/flr/doc/

- For more information on the FLR Project for Quantitative Fisheries Science in R, visit the FLR webpage, http://flr-project.org.
- You can submit bug reports, questions or suggestions specific to **FLBEIA** to flbeia@azti.es.

Software Versions

- R version 3.3.3 (2017-03-06)
- FLCore: 2.6.5
- FLBEIA: 1.15.2
- FLFleet: 2.6.0
- FLash: 2.5.0
- FLAssess: 2.6.1
- FLXSA: 2.5.20140808
- ggplotFL: 2.6.2
- ggplot2: 2.2.1
- **Compiled**: Wed Oct 04 09:15:44 2017

License

Author information

FLBEIA team AZTI. Marine Reserach Unit. Txatxarramendi Ugartea z/g, 48395, Sukarrieta, Basque Country, Spain. ** Mail** flbeia@azti.es