Package: mse (via r-universe)

September 5, 2024

Title Tools for Running Management Strategy Evaluations using FLR

Version 2.2.3.9266

Description A set of functions and methods to enable the development and running of Management Strategy Evaluation (MSE) analyses, using the FLR packages and classes and the a4a methods and algorithms.

X-schema.org-keywords simulation, MSE, fisheries, flr, a4a, R

Depends R(>= 4.0), methods, FLCore(>= 2.6.18), ggplotFL(>= 2.6.10), FLFishery, FLasher(>= 0.6.7), FLBRP(>= 2.5.8), doFuture, data.table, progressr

Imports parallelly, ggrepel, patchwork

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

Suggests testthat, knitr, rmarkdown, FLa4a, doParallel

Collate 'generics.R' 'mseCtrl-class.R' 'mpCtrl-class.R' 'oem.R' 'FLoem-class.R' 'FLo-class.R' 'FLom-class.R' 'FLombf-class.R' 'FLiem-class.R' 'FLmse-class.R' 'FLmses-class.R' 'dispatch.R' 'fb.R' 'hcr.R' 'iem.R' 'is.R' 'mp.R' 'phcr.R' 'ind.R' 'sa.R' 'tm.R' 'grid.R' 'mc.R' 'performance.R' 'run.R' 'tune.R' 'om.R' 'tracking.R' 'data.R' 'db.R' 'utilities.R' 'plot.R'

VignetteBuilder knitr

License EUPL

LazyLoad true

LazyData false

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

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Repository https://ices-tools-prod.r-universe.dev

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

RemoteRef HEAD

RemoteSha a3557278bf2e930248a01b842ab9d78113ffe6df

Contents

`bisect` *Bisection search for a forecast target providing a given performance statistic value.*

Description

The plain bisection algorithm (Burden & Douglas, 1985) is employed here to find the value of a given forecast target quantity (e.g. `fbar`) for which a selected value of a performance statistic is obtained over a chosen period.

Usage

```
bisect(
  stock,
  sr,
  deviances = rec(stock) %=% 1,
  metrics,
  refpts,
  statistic,
  years,
  pyears = years,
  tune,
  prob,
  tol = 0.01,
  maxit = 15,
  verbose = TRUE
)
```

References

Burden, Richard L.; Faires, J. Douglas (1985), "2.1 The Bisection Algorithm", Numerical Analysis (3rd ed.), PWS Publishers, ISBN 0-87150-857-5

Examples

```
data(ple4)
stock <- propagate(stf(ple4, end=2118), 100)
srr <- predictModel(model=rec~a*ssb*exp(-b*ssb), params=FLPar(a=5.20, b=1.65e-6))
# GENERATE SRR deviances
devs <- ar1rlnorm(rho=0.4, 2018:2118, iters=100, meanlog=0, sdlog=0.5)
# DEFINE Fp05 statistic
statistic <- list(FP05=list(~yearMeans((SB/SBlim) < 1), name="P.05",
  desc="ICES P.05"))
# CALL bisect over 100 years, Fp.05 calculated over last 50.
fp05fwd <- bisect(stock, sr=srr, deviances=devs, metrics=list(SB=ssb),
  refpts=FLPar(SBlim=150000), statistic=statistic, years=2018:2118,
  pyears=2069:2118, tune=list(fbar=c(0.1, 1)), prob=0.05)
```

---

catchSSB.hcr *A HCR to set total catch based on SSB depletion level*

Description

A HCR to set total catch based on SSB depletion level

Usage

```
catchSSB.hcr(
  stk,
  dtarget = 0.4,
  dlimit = 0.1,
  lambda = 1,
  MSY,
  dtaclow = 0.85,
  dtacupp = 1.15,
  yrs = 1,
  metric = "ssb",
  args,
  tracking
)
```

Arguments

stk	The perceived FLStock.
MSY	Assumed or estimated MSY.
args	MSE arguments, class *list*.
tracking	Structure for tracking modules outputs.
dtarget=0.40	Depletion level from which catch is decreased.
dlimit=0.10	Depletion level at which fishing is stopped.
lambda=1	Multiplier for MSY level.
dtaclow=0.85	Maximum proportional decrease in allowable catch.
dtacupp=1.15	Maximum proportional increase in allowable catch.

Value

A *list* with elements *ctrl*, of class *fwdControl*, and *tracking*.

Examples

```
data(sol274)
catchSSB.hcr(stock(om), MSY=140000, tracking=FLQuant(),
args=list(ay=2018, data_lag=1, management_lag=1, frq=1))
# APPLY hcr over a range of dtarget values
lapply(seq(0.30, 0.80, by=0.1), function(x) {
  catchSSB.hcr(stock(om), MSY=140000, dtarget=x,
  args=list(ay=2018, data_lag=1, management_lag=1, frq=1),
  tracking=FLQuant())$ctrl } )
```

computeFp05 *Calculates the Fbar value giving a maximum probability of ssb being below Blim of 5%*

Description

Calculates the Fbar value giving a maximum probability of ssb being below Blim of 5%

Usage

```
computeFp05(
  stock,
  sr,
  SBlim,
  range = c(0.01, 0.75),
  nyears = 3,
  sigmaR = 0.5,
  rho = 0.43,
  its = 500,
  verbose = TRUE
)
```

Arguments

stock	An FLStock over which the calculation is carried out.
sr	The stock-recruits relationship to use in fwd.
its	Number of iterations
verbose	Should progress be shown, TRUE.

Examples

```
data(ple4)
sr <- predictModel(model=bevholt, params=FLPar(a=1.4e6, b=1.5e5))
fp05 <- computeFp05(ple4, sr, SBlim=150000, its=300, range=c(0.40, 0.50))
# RUN projection for obtained Fp.05 value
proj <- fwd(propagate(stf(ple4, nyears=100), 300), sr=sr,
  fbar=FLQuant(fp05, dimnames=list(year=2018:2117)),
  deviances=ar1rlnorm(rho=0.43, years=2018:2117, iters=300, meanlog=0,
  sdlog=0.5))
plot(ssb(proj), prob=c(0.01, 0.25, 0.50, 0.75, 0.99)) +
  geom_hline(yintercept=150000)
```

cpue.hcr *cpue.hcr*

Description

cpue.hcr

Usage

```
cpue.hcr(
  stk,
  ind,
  k1 = 0.2,
  k2 = 0.2,
  k3 = 0.2,
  k4 = 0.2,
  target = 1,
  dtaclow = 0.85,
  dtacupp = 1.15,
  initac = NULL,
  slope = "slope",
  mean = "mean",
  args,
  tracking
)
```

Examples

```
data(sol274)
ind <- cpue.ind(stock(om), FLIndices(CPUE=FLIndexBiomass(index=ssb(om))),
  args=list(ay=2000, data_lag=1),
  tracking=FLQuant(dimnames=list(metric="ind", year=2000, iter=1:100)))
cpue.hcr(stk=stock(om), ind=ind$ind, k1=0.1, k2=0.2, k3=0.1, k4=0.1,
  args=list(ay=2000, frq=1, management_lag=1),
  tracking=FLQuants(sol174=FLQuant(1000, dimnames=list(metric="hcr",
  year=2000))))
```

db *Creates a single table of output from an FLmse object*

Description

Creates a single table of output from an FLmse object

Usage

```
db(object, ...)
```

Arguments

object	Input object with MSE results
metrics	List of metrics to be computed on relevant slots

Value

A data.table object

---

debug-mse *Debugging mse modules*

Description

Set and unset the debugging flag of a function inside the *method* slot of a mseCtrl object.

Usage

```
debug(fun, text = "", condition = NULL, signature = NULL)

## S4 method for signature 'mseCtrl,missing'
debug(fun)

## S4 method for signature 'mseCtrl,missing'
undebug(fun)

## S4 method for signature 'mpCtrl,character'
undebug(fun, signature = NULL)

## S4 method for signature 'mpCtrl,missing'
undebug(fun)

## S4 method for signature 'FLo,ANY'
debug(fun)

## S4 method for signature 'FLo,ANY'
undebug(fun)
```

Arguments

fun	Module or control object to debug.
text	Name of module in mpCtrl.
condition	Unused.
signature	Name of module in mpCtrl.

Details

Modules in the mse control object contain the function to be called in the *method* slot. To debug and check the behaviour of an individual function, the *debug* method will start a browser session next time it is called. Debugging functions requires the parallel flag to be set to FALSE, or that no parallel backend is loaded.

Calling *undebug* on an mpCtrl without specifying a module will check for the debugging status of each of them, and undebug if TRUE.

For objects of classes *FLom* and *FLombf*, *debug* and *undebug* will set and unset the debugging flag on the function stored in the *projection* slot.

Value

Both functions invisibly return NULL

Author(s)

Iago Mosqueira (WMR)

See Also

debug

`effort.is` *effort implementation function*

Description

effort implementation function

Usage

```
effort.is(stk, ctrl, args, tracking)
```

Arguments

`stk`	The perceived FLStock.
`imp_control`	A list with the elements: nsqy, delta_tac_min, delta_tac_max
`ay`	The year for which the target F is set, based on the SSB in year (ay - control$ssb_lag).

fixedC.hcr *A fixed catch HCR*

Description

No matter what get C = ctrg The control argument is a list of parameters used by the HCR.

Usage

```
fixedC.hcr(stk, ctrg, args, tracking)
```

Arguments

stk	The perceived FLStock.
control	A list with the element ctrg (numeric).

Examples

```
data(sol274)
fixedC.hcr(stock(om), ctrg=50000, args=list(ay=2017, management_lag=1,
  frq=1), tracking=FLQuant())
```

fixedF.hcr *A fixed target f*

Description

No matter what get F = Ftarget The control argument is a list of parameters used by the HCR.

Usage

```
fixedF.hcr(stk, ftrg, args, tracking)
```

Arguments

stk	The perceived FLStock.
control	A list with the element ftrg (numeric).

Examples

```
data(sol274)
fixedF.hcr(stock(om), ftrg=0.15, args=list(ay=2017, management_lag=1,
  frq=1), tracking=FLQuant())
```

FLiem-class *S4 class* FLiem

Description

The FLiem class stores the information relative to the implementation error model of the MSE.

Usage

```
## S4 method for signature 'FLiem'
initialize(.Object, ...)

## S4 method for signature 'FLiem'
iter(obj, iter)
```

Arguments

...	additional argument list that might never be used
object	object of relevant class (see signature of method)

Slots

method character with the name of the method to be run. Note a function of method must exist in the environment with the same name.

args list with arguments to be passed to the function defined in method

Accessors

All slots in the class have accessor and replacement methods defined that allow retrieving and substituting individual slots.

The values passed for replacement need to be of the class of that slot. A numeric vector can also be used when replacing FLQuant slots, and the vector will be used to substitute the values in the slot, but not its other attributes.

Constructor

A construction method exists for this class that can take named arguments for any of its slots. All slots are then created to match the requirements of the class validity. If an unnamed FLQuant object is provided, this is used for sizing, but not for populating any slot.

FLmse-class *S4 class* FLmse

Description

The FLmse class stores information relative to the MSE's management procedure'.

Usage

```
FLmse(...)

FLmse(...)

om(object, ...)

## S4 method for signature 'FLmse'
om(object)

om(object) <- value

tracking(object, ...)

## S4 method for signature 'FLmse'
tracking(object, biol = "missing")

tracking(object, ...) <- value

## S4 method for signature 'FLmse'
control(object, i = "missing")

oem(object, ...)

## S4 method for signature 'FLmse'
oem(object)

oem(object) <- value

## S4 method for signature 'FLmse'
args(name)

## S4 replacement method for signature 'FLmse,list'
args(object) <- value
```

Arguments

...	additional argument list that might never be used
object	object of relevant class (see signature of method)

value the new object

Slots

om FLom with the operating model.

tracking FLQuant with record of decisions made during the mp cycle.

args list with assorted arguments required to run the MSE cycle.

Accessors

All slots in the class have accessor and replacement methods defined that allow retrieving and substituting individual slots.

The values passed for replacement need to be of the class of that slot. A numeric vector can also be used when replacing FLQuant slots, and the vector will be used to substitute the values in the slot, but not its other attributes.

Constructor

A construction method exists for this class that can take named arguments for any of its slots. All slots are then created to match the requirements of the class validity. If an unnamed FLQuant object is provided, this is used for sizing, but not for populating any slot.

FLoem-class *Specification for the observation error model (OEM).*

Description

The FLoem class stores the method, arguments and observations that define the way observations are collected from an operating model at each time step in the management procedure. This class extends *mseCtrl* through the addition of two lists used to gather past and new observations, and deviances to use on each step in the observation process.

Usage

```
FLoem(...)

FLoem(...)

observations(object, ...)

## S4 method for signature 'FLoem'
observations(object, ...)

observations(object, i) <- value

## S4 replacement method for signature 'FLoem,missing,list'
observations(object) <- value
```

```
## S4 replacement method for signature 'FLoem,ANY,FLStock'
observations(object, i) <- value

## S4 method for signature 'FLoem'
deviances(object, ...)

## S4 replacement method for signature 'FLoem,list'
deviances(object, ...) <- value

## S4 method for signature 'FLoem'
show(object)

## S4 method for signature 'FLoem'
iter(obj, iter)

## S4 method for signature 'FLoem,FLoem'
combine(x, y, ..., check = FALSE)
```

Arguments

`...`	additional argument list that might never be used
`object`	object of relevant class (see signature of method)
`value`	the new object

Slots

`method` The function to be run in the module call, class *function*.

`args` Arguments to be passed to the method, of class *list*.

`observations` Past observations, class *list*.

`deviances` Observation deviances, class *list*.

Accessors

All slots in the class have accessor and replacement methods defined that allow retrieving and substituting individual slots.

The values passed for replacement need to be of the class of that slot. A numeric vector can also be used when replacing FLQuant slots, and the vector will be used to substitute the values in the slot, but not its other attributes.

Constructor

A construction method exists for this class that can take named arguments for any of its slots. All slots are then created to match the requirements of the class validity. If an unnamed `FLQuant` object is provided, this is used for sizing, but not for populating any slot.

Examples

```
data(sol274)
deviances(oem)
deviances(oem, "stk")
deviances(oem, "stk", "catch.n")
```

FLom *A class for an operating model (OM)*

Description

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque eleifend odio ac rutrum luctus. Aenean placerat porttitor commodo. Pellentesque eget porta libero. Pellentesque molestie mi sed orci feugiat, non mollis enim tristique. Suspendisse eu sapien vitae arcu lobortis ultrices vitae ac velit. Curabitur id

Usage

```
FLom(...)

FLom(...)

## S4 method for signature 'FLom'
stock(object)

## S4 replacement method for signature 'FLom,FLStock'
stock(object) <- value

## S4 method for signature 'FLom'
sr(object)

## S4 replacement method for signature 'FLom,FLSR'
sr(object) <- value

## S4 method for signature 'FLom,FLom'
combine(x, y, ...)

## S4 replacement method for signature 'FLmse,FLo'
om(object) <- value

## S4 replacement method for signature 'FLmse,FLQuants'
tracking(object) <- value

## S4 replacement method for signature 'FLmse,mpCtrl'
control(object) <- value

## S4 replacement method for signature 'FLmse,FLoem'
oem(object) <- value
```

Arguments

`...`	additional argument list that might never be used
`object`	object of relevant class (see signature of method)
`value`	Object to assign in slot

Slots

`stock` The population and catch history, `FLStock`.

`sr` The stock-recruitment relationship, `FLSR`.

`refpts` The estimated reference points, `FLPar`.

`fleetBehaviour` Dynamics of the fishing fleet to be used in projections, `mseCtrl`.

Validity

stock and sr dimensions Dimensions 2:6 of the `stock` and `sr` slots must match.

rec age Stock and stock recruitment residuals must use the recruitment age.

You can inspect the class validity function by using `getValidity(getClassDef('FLom'))`

Accessors

All slots in the class have accessor and replacement methods defined that allow retrieving and substituting individual slots.

The values passed for replacement need to be of the class of that slot. A numeric vector can also be used when replacing FLQuant slots, and the vector will be used to substitute the values in the slot, but not its other attributes.

Constructor

A construction method exists for this class that can take named arguments for any of its slots. All unspecified slots are then created to match the requirements of the class validity function.

Methods

Methods exist for various calculations based on values stored in the class:

METHOD Neque porro quisquam est qui dolorem ipsum.

Author(s)

The FLR Team

See Also

FLComp

Examples

```
data(sol274)
comb <- combine(iter(om, 1:50), iter(om, 51:100))
all.equal(om, comb)
```

`fwd.om` *A method to project the operating model (OM)*

Description

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque eleifend odio ac rutrum luctus. Aenean placerat porttitor commodo. Pellentesque eget porta libero. Pellentesque molestie mi sed orci feugiat, non mollis enim tristique. Suspendisse eu sapien vitae arcu lobortis ultrices vitae ac velit. Curabitur id

Usage

```
fwd.om(om, ctrl, ...)
```

Arguments

`ctrl`	the fwdControl object with objectives and constraints
`...`	
`object`	the OM as a FLStock

`grid` *Functions to create grids and lists of model and simulation runs*

Description

Grids of combinations of variables (or parameters) can be created from a list of scenarios contaning all possible values for each variables. Such a grid can be used when conditioning an operating model around the structural uncertainty of fixed parameters and submodel choices. And it can also be used to create a range of combinations of management procedure parameters for *tuning* and selection of MPs.

Usage

```
expandGrid(scenarios, ..., names = FALSE)

gridList(scenarios, ..., grid = missing)
```

Arguments

scenarios	Values and levels for each run variable, *list*.
...	Extra arguments, see *Details*.
names	Should the table output contain names (FALSE) or indices (TRUE, default).
grid	Index table with grid to be converted into input list.

Details

The list of *scenarios* used as input should contain one element per variable, be it either a vector or a list of alternative values. The list must be named, and elements of class *list* should themselves be named. Names are added to vector elements by calling *as.character()* on their content. See examples below for guidance.

Value

A data.frame or list.

Author(s)

Iago Mosqueira, EC JRC

See Also

FLComp

Examples

```
# Lists of scenarios can contains vectors ...
list(steepness=c(0.6, 0.7, 0.8), M=c(0.2, 0.3, 0.4))
# lists ...
list(M=list(0.2, 0.4, lo=seq(0.2, 0.4, length=10)))
# or both
scenarios  <- list(steepness=c(0.7, 0.8), M=list(0.2, 0.4, lo=seq(0.2, 0.4, length=10)))
# Create full grid as index table
expandGrid(scenarios)
# Drop certain combinations
expandGrid(scenarios, M == 0.2 & steepness == 0.7, M == "lo" & steepness == 0.8)
# Output as names
expandGrid(scenarios, M == 0.2 & steepness == 0.7, names=TRUE)
# Create list of variable combinations
runs <- gridList(scenarios)
runs[[1]]
length(runs)
# Create list but dropping certain combinations
runs <- gridList(scenarios, M == 0.2 & steepness == 0.7)
runs[[1]]
length(runs)
```

`ices.hcr` *The typical HCR used by ICES*

Description

The typical HCR used by ICES which sets a target F based on the SSB based on 4 parameters: sblim, sbsafe, fmin and ftrg. F increases linearly between SSB = blim and SSB = bsafe, from F = fmin to F = ftrg. If:

- B < Blim, F = Fbycatch;
- B > trigger, F = Fmsy;
- B > Blim & B < trigger, F linear between Fbycatch and Fmsy;
- F = ftrg is the maximum F, F = fmin is the minimum F. F is set in year ay, based on SSB in year ay - data_lag

Usage

```
ices.hcr(
  stk,
  ftrg,
  sblim,
  sbsafe,
  fmin = 0,
  minfbar = range(stk, "minfbar"),
  maxfbar = range(stk, "maxfbar"),
  args,
  tracking
)
```

Arguments

`stk`	The perceived FLStock.
`ftrg`	TODO:description
`sblim`	TODO:description
`sbsafe`	TODO:description
`fmin`	Minimum fishing mortality.
`args`	MSE arguments, class *list*.
`tracking`	Structure for tracking modules outputs.

Value

A *list* with elements *ctrl*, of class *fwdControl*, and *tracking*.

Examples

```
data(ple4)
# Test for year when SSB > bsafe
ices.hcr(ple4, fmin=0.05, ftrg=0.15, sblim=200000, sbsafe=300000,
  args=list(ay=2018, data_lag=1, management_lag=1), tracking=FLQuant())
# Test for year when SSB < bsafe
ices.hcr(ple4, fmin=0.05, ftrg=0.15, sblim=200000, sbsafe=300000,
  args=list(ay=1995, data_lag=1, management_lag=1), tracking=FLQuant())
```

---

index.hat,FLIndexBiomass,FLStock-method *Predicted index of abundance from abundance estimates*

Description

Predicted index of abundance from abundance estimates

Usage

```
## S4 method for signature 'FLIndexBiomass,FLStock'
index.hat(object, stock)
```

Arguments

object	An FLIndexBiomass object.
stock	An FLStock object with estimated abundances, *stock.n*.

Value

An FLQuant for the precited index of abundance in biomass.

---

indicator.hcr *An indicator-based HCR*

Description

Get indicator to target. The control argument is a list of parameters used by the HCR.

Usage

```
indicator.hcr(stk, hcrpars, args, tracking)
```

Arguments

stk	The perceived FLStock.
args	A list with generic arguments to be used by the function if needed.
tracking	The tracking matrix.
itrg	The target for the indicator.

indicator.is *indicator implementation function*

Description

indicator implementation function

Usage

```
indicator.is(stk, ctrl, args, tracking, system = c("output", "input"), ...)
```

Arguments

stk	The perceived FLStock.
ctrl	control file with HCR decision

initiate *Initializes a population for a given virgin biomass.*

Description

Abundances at age for a population at virgin conditions at age. An FLBiol object is initiated by providing a target total biomass (B0) and a value for the stock-recruit steepness (s). The object requires slots to be already filled up for the mean weight-at-age (wt), natural mortality (m), time of spawning (spwn) and maturity at age (mat).

Usage

```
initiate(biol, B0, h = 0.75)
```

Arguments

biol	An FLBiol object to nbe initiated.
B0	Initial or virgin biomass.

Value

An updated FLBiol with abundances set in teh first year to match the requested biomas.

Examples

```
data(ple4.biol)
initiate(ple4.biol, B0=450000)
#
initiate(propagate(ple4.biol, 100), B0=runif(100, 3e5, 5e5))
```

kobestatistics *Kobe statistics*

Description

Aliquam sagittis feugiat felis eget consequat. Praesent eleifend dolor massa, vitae faucibus justo lacinia a. Cras sed erat et magna pharetra bibendum quis in mi. Sed sodales mollis arcu, sit amet venenatis lorem fringilla vel. Vivamus vitae ipsum sem. Donec malesuada purus at libero bibendum accumsan. Donec ipsum sapien, feugiat blandit arcu in, dapibus dictum felis.

Format

An object of class list.

mcN *Compute number of necessary Monte Carlo runs*

Usage

```
mcN(x, ...)
```

Arguments

z	Value of Z for a given confidence level for a normally distributed random variable, default is 1.96 for a 95\
	\itemEThe required percentage error of the mean, *numeric*.

The mcN function implements the simple method of Bukaci et al. (2016) to calculate the number of Monte Carlo (MC) simulations required to obtain results with a given precision level.

data(ple4) ssb <- rlnorm(2000, log(ssb(ple4)), 0.5) itse <- mcN(ssb) plot(se~iters, itse, type='l')

Bukaci, E. Korini, Th., Periku, E., Allkja, S., Sheperi, P. 2016. Number of iterations needed in Monte Carlo Simulation using reliability analysis for tunnel supports. Int. J. of Eng. Res. and Apps., 6 (6-3): 60-64. <www.ijera.com/papers/Vol6_issue6/Part%20-%203/J0606036064.pdf>

movingF.hcr *TODO:description*

Description

TODO:description

Usage

```
movingF.hcr(stk, hcrpars, args, tracking)
```

Arguments

stk	TODO:description
hcrpars	TODO:description
args	TODO:description
tracking	TODO:description

Value

TODO:description

mp *mp executes a single run of a Management Procedure*

Description

An individual management procedure (MP) is run for a number of years, on an operating model, observation error model, control and arguments.

Usage

```
mp(
  om,
  oem = NULL,
  iem = NULL,
  control = ctrl,
  ctrl = control,
  args,
  scenario = "NA",
  tracking = "missing",
  logfile = tempfile(),
  verbose = !handlers(global = NA),
  parallel = TRUE
)
```

Arguments

om	The operating model (OM), an object of class *FLom* or *FLombf*.
oem	The observation error model (OEM), an object of class *FLoem*.
iem	The implementation error model (IEM), an object of class *FLiem*.
ctrl	A control structure for the MP run, an object of class *mpCtrl*.
args	MSE arguments, *list*. Only 'iy', the intermediate (starting) year, is required.
scenario	Name of the scenario tested in this run, *character*.
tracking	Extra elements (rows) to add to the standard tracking *FLQuant* in its first dimensions, *character*.
verbose	Should output be verbose or not, *logical*.

Value

An object of class *FLmse*.

Examples

```
# dataset contains both OM (FLom) and OEM (FLoem)
data(sol274)
# Set control: sa and hcr
control <- mpCtrl(list(
  est = mseCtrl(method=perfect.sa),
  hcr = mseCtrl(method=hockeystick.hcr, args=list(lim=0,
  trigger=41500, target=0.27))))
tes <- mp(om, oem=oem, ctrl=control, args=list(iy=2021, fy=2034))
tes3 <- mp(om, oem=oem, ctrl=control, args=list(iy=2021, fy=2034, frq=3))
plot(om, list(annual=tes, triannual=tes3))
# 'perfect.oem' is used if none is given
tes <- mp(om, ctrl=control, args=list(iy=2021, fy=2035))
plot(om, tes)
```

---

mpCtrl-class *S4 class* mpCtrl

Description

The mpCtrl class defines which modules will be run my a call to the mp function. It contains a series of objects of class *mseCtrl* only for those modules required by the defined MP.

Usage

```
## S4 method for signature 'mpCtrl'
initialize(.Object, ...)

est(object, ...)
```

```
## S4 method for signature 'mpCtrl'
est(object)

est(object) <- value

## S4 replacement method for signature 'mpCtrl,function'
est(object) <- value

phcr(object, ...)

## S4 method for signature 'mpCtrl'
phcr(object)

phcr(object) <- value

## S4 replacement method for signature 'mpCtrl,function'
phcr(object) <- value

hcr(object, ...)

## S4 method for signature 'mpCtrl'
hcr(object)

hcr(object) <- value

## S4 replacement method for signature 'mpCtrl,function'
hcr(object) <- value

isys(object, ...)

## S4 method for signature 'mpCtrl'
isys(object)

isys(object) <- value

## S4 replacement method for signature 'mpCtrl,function'
isys(object) <- value

tm(object, ...)

## S4 method for signature 'mpCtrl'
tm(object)

tm(object) <- value

## S4 replacement method for signature 'mpCtrl,function'
tm(object) <- value
```

```
## S4 method for signature 'mpCtrl'
show(object)

## S4 method for signature 'mpCtrl'
iters(object, iter)

## S4 method for signature 'mpCtrl'
iter(obj, iter)

## S4 method for signature 'mpCtrl'
method(object, element)

## S4 replacement method for signature 'mpCtrl,function'
method(object, element) <- value

## S4 replacement method for signature 'mpCtrl,function'
args(object, element) <- value

## S4 method for signature 'mpCtrl,character'
debug(fun, text)
```

Arguments

`...`	additional argument list that might never be used
`object`	object of relevant class (see signature of method)

Slots

`est` Specification for the stock status estimator, class *mseCtrl*.

`phcr` Specification for the harvest control rule parametrization, class *mseCtrl*.

`hcr` Specification for the harvest control rule, class *mseCtrl*.

`isys` Specification for the implementation system, class *mseCtrl*.

`tm` Specification for technical measures, class *mseCtrl*.

Accessors

All slots in the class have accessor and replacement methods defined that allow retrieving and substituting individual slots.

The values passed for replacement need to be of the class of that slot. A numeric vector can also be used when replacing FLQuant slots, and the vector will be used to substitute the values in the slot, but not its other attributes.

Constructor

A construction method exists for this class that can take named arguments for any of its slots. All slots are then created to match the requirements of the class validity. If an unnamed `FLQuant` object is provided, this is used for sizing, but not for populating any slot.

Examples

```
mpCtrl(list(
  est = mseCtrl(method=perfect.sa),
  hcr = mseCtrl(method=hockeystick.hcr, args=list(lim=0,
  trigger=41500, target=0.27))))
mpCtrl(list(
  est = mseCtrl(method=perfect.sa),
  hcr = mseCtrl(method=hockeystick.hcr, args=list(lim=0,
  trigger=41500, target=0.27))))
```

---

mseCtrl-class *S4 class* mseCtrl

Description

The mseCtrl class stores information about how a specific module will be run. The function contained in the *method* slot will be called with three sets of argument: those contained in the *args* slot, the *args* argument to the call to the *mp* function, and the inputs defined by type of module being defined by a particular object. Please see the "Module dispatch" section in the *mse* Technical Manual.

Usage

```
## S4 method for signature 'mseCtrl'
initialize(.Object, ..., method, args)

method(object, ...)

## S4 method for signature 'mseCtrl'
method(object)

method(object, ...) <- value

## S4 replacement method for signature 'mseCtrl,function'
method(object) <- value

args(name)

## S4 method for signature 'mseCtrl'
args(name)

args(object, ...) <- value

## S4 replacement method for signature 'mseCtrl,list'
args(object) <- value

## S4 method for signature 'mseCtrl'
```

```
show(object)

exists(
  x,
  where = -1,
  envir = if (missing(frame)) as.environment(where) else sys.frame(frame),
  frame,
  mode = "any",
  inherits = TRUE
)

## S4 method for signature 'mseCtrl'
exists(x)

## S4 method for signature 'mpCtrl'
exists(x)
```

Arguments

`...`	additional argument list that might never be used
`object`	object of relevant class (see signature of method)

Slots

`method` The function to be run in the module call, class *function*.

`args` Arguments to be passed to the method, of class *list*.

Accessors

All slots in the class have accessor and replacement methods defined that allow retrieving and substituting individual slots.

The values passed for replacement need to be of the class of that slot. A numeric vector can also be used when replacing FLQuant slots, and the vector will be used to substitute the values in the slot, but not its other attributes.

Constructor

A construction method exists for this class that can take named arguments for any of its slots. All slots are then created to match the requirements of the class validity. If an unnamed `FLQuant` object is provided, this is used for sizing, but not for populating any slot.

Examples

```
ctl <- mseCtrl(method=function(stk, args, alpha) ssb(stk) * alpha,
  args=list(alpha=0.5))
ctl
method(ctl)
method(ctl) <- function(stk, args, beta) ssb(stk) * beta
args(ctl)
```

```
args(ctl) <- list(beta=0.9)
exists(ctl)
```

---

p4om *FLom object for North sea plaice*

Description

Aliquam sagittis feugiat felis eget consequat. Praesent eleifend dolor massa, vitae faucibus justo lacinia a. Cras sed erat et magna pharetra bibendum quis in mi. Sed sodales mollis arcu, sit amet venenatis lorem fringilla vel. Vivamus vitae ipsum sem. Donec malesuada purus at libero bibendum accumsan. Donec ipsum sapien, feugiat blandit arcu in, dapibus dictum felis.

Format

An object of class *FLom*.

---

perfect.oem *A perfect observation of catch and abundances-at-age.*

Description

This observation error model function generates a set of perfect observations on catches, biology and abundance. Direct observations are made of the stock, while a single age-structured index of abundance, in numbers, is created with a fixed catchability of 0.01. *deviances* on either *stk$catch.n* or *idx$index*, if given, are applied.

Usage

```
perfect.oem(stk, deviances, observations, args, tracking, biomass = FALSE, ...)
```

Arguments

deviances	A named list of observation deviances, class *list*.
observations	A list of past observations, extended to the end of *om*, class *list*.
args	The mp dimensions arguments, as generated by mp, class *list*.
tracking	Object to track module decisions and outputs, class *FLQuant*.
om	An operating model, class *FLom* or *FLombf*.

Details

This *oem* function generates a full observation time series every time step, and does not append them to existing objects in *observations*.

Value

A named *list* with elements *stk* (*FLStock*), *idx* (*FLIndices*), *deviances*, *observations* and *tracking*.

Examples

```
# On FLom
data(sol274)
obs <- perfect.oem(stock(om), deviances=NULL, observations=NULL,
  args=list(y0=1957, dy=2021), tracking=FLQuant())
```

---

perfect.sa *A perfect 'estimate' of abundances, catches and harvest.*

Description

The *FLStock* generated by the call to *oem* is simply passed on in this function. The estimates of abundance, catches and exploitation will thus be as precise as the OEM observation.

Usage

```
perfect.sa(stk, idx, args, tracking, ...)
```

Arguments

stk	The stock observation generated by *oem*. Class *FLStock*.
idx	An observation of chnages in abundance, not used. Class *FLIndices*.
args	MSE arguments, class *list*.
tracking	Structure for tracking modules outputs.

Value

A *list* with elements *stk* and *tracking*.

Examples

```
data(sol274)
perfect.sa(stock(om), FLIndices(), args=list(ay=2018, dy=2017),
  tracking=FLQuants(FLQuant(dimnames=list(metric="conv.est", year=2018))))
```

`performance` *Compute performance statistics*

Description

TODO

Usage

```
## S4 method for signature 'FLQuants'
performance(
  x,
  statistics,
  refpts = FLPar(),
  years = setNames(list(dimnames(x[[1]])$year), nm = dims(x[[1]])$maxyear),
  probs = c(0.1, 0.25, 0.5, 0.75, 0.9),
  mp = NULL
)

## S4 method for signature 'FLStock'
performance(
  x,
  statistics,
  refpts = FLPar(),
  years = as.character(seq(dims(x)$minyear, dims(x)$maxyear)),
  metrics = FLCore::metrics(x),
  probs = NULL,
  mp = NULL
)

## S4 method for signature 'FLStocks'
performance(
  x,
  statistics,
  refpts = FLPar(),
  years = dims(x[[1]])$maxyear,
  metrics = FLCore::metrics,
  probs = NULL,
  grid = missing,
  mp = NULL,
  mc.cores = 1
)

## S4 method for signature 'list'
performance(
  x,
  statistics,
```

```
  refpts = FLPar(),
  years = dims(x[[1]])$maxyear,
  probs = NULL,
  grid = "missing",
  mp = NULL,
  mc.cores = 1,
  ...
)

## S4 method for signature 'FLom'
performance(x, refpts = x@refpts, metrics = NULL, statistics = NULL, ...)
```

Arguments

statistics	statistics to be computed, as formula, name and description, list
refpts	Reference points for calculations, list
years	Years on which statistics should be computed, defaults to last year of input FLQuants
run	Object holding the results of forward projections, as a named FLQuants

Details

Each statistics is an object of class list object, with three elements, the first two of them compulsory:

- An unnamed element of class *formula*, e.g. `yearMeans(SB/SB0)`.
- name: A short name to be output on tables and plots, of class character, e.g. "SB/SB0".
- desc: A longer description of the statistics, of class character, e.g. "Mean spawner biomass relative to unfished"

Each statistic `formula` is evaluated against the *metrics* and *refpts* used in the function call. Formulas can thus use (i) the names of the `FLQuants` object or of the object returned by the call to `metrics()`, (ii) of the *params* in the *refpts* object and, for all classes but `FLQuants`, (iii) functions that can be called on *object*. See examples below for the necessary matching between *metrics*, *refpts* and the statistics formulas.

Value

data.table Results of computing performance statistics.

Author(s)

Iago Mosqueira, EC JRC

See Also

FLQuants

Examples

```
# LOAD example FLmse object
data(sol274)
# GENERATE pseudo-run from last 20 years of OM
run <- window(stock(om), start=2012, end=2021)
# DEFINE statistics
statistics <- list(
  dCatch=list(~yearMeans(C[, -1]/C[, -dims(C)$year]),
    name="mean(C[t] / C[t-1])",
    desc="Mean absolute proportional change in catch"),
  varCatch=list(~yearVars(C),
    name="var(C)",
    desc="Variance in catch"),
  varF=list(~yearVars(F),
    name="var(F)",
    desc="Variance in fishing mortality"))
# COMPUTE performance
performance(run, statistics, refpts=FLPar(MSY=110000),
  metrics=list(C=catch, F=fbar), years=list(short=2016:2018, long=2016:2021))
# Minimum statistic, named list with formula and name
performance(run, statistics=list(CMSY=list(~yearMeans(C/MSY), name="CMSY")),
  refpts=FLPar(MSY=110000), metrics=list(C=catch, F=fbar),
  years=list(2012:2021))
# return quantiles
performance(run, statistics, refpts=FLPar(MSY=110000),
  metrics=list(C=catch, F=fbar), years=list(2012:2021),
  probs =  c(0.05, 0.1, 0.25, 0.5, 0.75, 0.9, 0.95))
# DEFINE statistics without summaries
statistics <- list(
  CMSY=list(~yearMeans(C/MSY),
    name="CMSY",
    desc="Catch over MSY"))
# COMPUTE performance
perf <- performance(run, statistics, refpts=FLPar(MSY=110000),
  metrics=list(C=catch), years=list(2012:2021))
# COMPUTE summaries
perf[, .(CMSY=mean(data))]
perf <- performance(FLStocks(B=run, A=run), statistics,
  refpts=FLPar(MSY=110000), metrics=list(C=catch), years=list(2012:2015))
```

---

sampling.oem *Samples from an operating model to obtain catch, biology and abundance data*

Description

This observation error model (OEM) function mimics the most common data collection regime, in which catch-at-age and biology is sampled from the population, and one or more indices of abundance are derived from surveys or CPUE data.

Usage

```
sampling.oem(stk, deviances, observations, stability = 1, args, tracking)
```

Arguments

stk	An FLStock object as obtained by the call to *stock(om)*.
deviances	A named list of deviances, see Details.
observations	A named list of observations, see Details.
args	Options and arguments passed on by *mp()*.
tracking	The tracking object.

Details

The FLStock object passed to *sampling.oem* by the *mp* function is simplified to match the dimensions of that present in the *observations* slot.

Value

A named list with elements *stk*, *idx*, *observations* and *tracking*.

Author(s)

Iago Mosqueira (WUR) & Ernesto Jardim (MSC).

See Also

mp

Examples

```
data(sol274)
# Generate samples from year 2000:2016
sampling.oem(stock(om), deviances=deviances(oem),
  observations=observations(oem),
  args=list(y0=2000, dy=2021, frq=1), tracking=FLQuant())
```

---

statistics *Example set of performance statistics*

Description

A set of performance statistics is provided, coded in the format that `performance()`requires. The statistics included in this list are as follows:

Format

An object of class list.

Details

- SB0: Mean spawner biomass relative to unfished.
- SBMSY: Mean spawnwer biomass relative to SBMSY.
- Ftarget: Mean fishing mortality relative to target.
- FMSY: Mean fishing mortality relative to FMSY.
- green: Probability of being in Kobe green quadrant.
- red: Probability of being in Kobe red quadrant.
- PSBMSY: Probability of SB greater or equal to SBMSY.
- PBlim: Probability that spawner biomass is above Blim.
- risk1: ICES Risk 1, mean probability that spawner biomass is below Blim.
- risk2: ICES Risk 2, probability that spawner biomass is above Blim once.
- risk3: ICES Risk 3, max probability that spawner biomass is above Blim.
- C: Mean catch over years.
- VarC: Catch variability.
- PC0: Probability of fishery shutdown.

Each indicator is itself a list object, with three elements, the first two of them compulsory:

- An unnamed element of class *formula*, e.g. `yearMeans(SB/SB0)`.
- name: A short name to be output on tables and plots, of class character, e.g. "SB/SB0".
- desc: A longer description of the indicator, of class character, e.g. "Mean spawner biomass relative to unfished"

`tac.is` *TAC implementation system module*

Description

Performs a short term forecast (STF) for the target fishing mortality to obtain the corresponding catch.

Usage

```
tac.is(
  stk,
  ctrl,
  args,
  output = "catch",
  recyrs = -2,
  Fdevs = fbar(fut) %=% 1,
  dtaclow = NA,
  dtacupp = NA,
  fmin = 0,
```

```
  reuse = TRUE,
  initac = metrics(stk, output)[, ac(iy - 1)],
  tracking
)
```

Arguments

stk	The perceived FLStock.
ctrl	The fwdControl output by the *hcr* step, target must be 'fbar'.
args	The MSE run arguments.
recyrs	Years to use for geometric mean recruitment if projection. Defaults to all years minus the last two.
Fdevs	Deviances on the fbar input to incorporate error and bias when MP is run using the pseudo-estimators 'perfect.sa' or 'shortcut.sa'.
dtaclow	Limit to decreases in output catch, as a proportional change (0.85 for 15%). Applied only when metric > lim, as set by 'hcr' step.
dtacupp	Limit to increases in output catch, as a proportional change (1.15 for 15%). Applied only when metric > lim, as set by 'hcr' step.
fmin	Minimum fbar to apply when catch change limits are use.
initac	Initial catch from which to compute catch change limits. Defaults to previous observed catch.
tracking	The tracking object.

Details

A `fwdControl` object obtained from the 'hcr' step is applied in the management year (`ay + mlag`) or years (`seq(ay + mlag, ay + mlag + freq)`. An assumption is made on the mortality in the assessment year (`ay`), which becomes the intermediate year in this projection. By default this is set to Fbar = Fsq, that is, the same fishing mortality estimated in the last data year (`ay - data_lag`).

The projection applies a constant recruitment, equal to the geometric mean over an specified number of years. By default all years minus the last two are included in the calculation. An specific set of years can be employed, by specifying a character vector of year names, or two values can be given for the number of years to be inlcuded, counting from the last, and how many years to exclude at the end. For example, `c(30, 2)` will use the last 30 years but excluding the last two, usually worst estimated.

Examples

```
data(sol274)
# Setup control with tac.is
control <- mpCtrl(list(est=mseCtrl(method=perfect.sa),
  hcr=mseCtrl(method=hockeystick.hcr,
    args=list(lim=0, trigger=4.3e5, target=0.21)),
  isys=mseCtrl(method=tac.is, args=list(recyrs=-3, output='landings'))))
# Run MP until 2025
run <- mp(om, oem, ctrl=control, args=list(iy=2021, fy=2027))
# Plot run time series
plot(om, TAC.IS=run)
```

target.hcr *Target-based harvest control rule to adjust input or output from CPUE*

Description

Short description

Usage

```
target.hcr(
  ind,
  lim,
  target,
  r = 1,
  metric = "mlc",
  output = "catch",
  nyears = 3,
  args,
  tracking
)
```

Value

A list containing *ctrl*, a `fwdControl` object, and *tracking*, an `FLQuant`.

Author(s)

The FLR Team

References

Hoshino, E., Hillary, R., Davies, C., Satria, F., Sadiyah, L., Ernawati, T., and Proctor, C. 2020. Development of pilot empirical harvest strategies for tropical tuna in indonesian archipelagic waters: case studies of skipjack and yellowfin tuna. *Fisheries Research*, 227:105539, doi:10.1016/j.fishres.2020.105539.

See Also

FLComp

Examples

```
data(sol274)
#
est <- cpue.ind(stock(om), FLIndices(CPUE=FLIndexBiomass(index=ssb(om))),
  args=list(ay=2000, data_lag=1),
  tracking=FLQuant(dimnames=list(metric="ind", year=2000, iter=1:100)))
#
target.hcr(ind=est$ind, lim=28000, target=40000,
```

```
metric="wmean", output="catch",
args=list(ay=2000, frq=1, data_lag=1, management_lag=1),
tracking=FLQuants(sol174=FLQuant(1000, dimnames=list(metric="hcr",
year=2000))))
```

---

tune *tune*

Description

Carry out multiple runs of an MP for a given dataset over a grid of values for the MP/HCR paramaters, in order to find the parameter combination(s) that give the best performance over the chosen statistics, a.k.a. *tuning*

Usage

```
tune(mp, grid, statistics, refpts, ...)
```

Arguments

mp	A function executing a projection applying a given MP, see mseBasic for an example
grid	A name list of *mp* argument values to loop along
statistics	A list of performance statistics
refpts	The reference points needed to compute the statistics, *FLPar*
...	Any other arguments to be passed on to *mp*

Details

DETAILS

Value

A list or aggregatecd FLR object, depending on the output of *mp*

See Also

mp,performance

Index