Package: FLSAM (via r-universe)

August 31, 2024

Title An Implementation of the State-Space Assessment Model for FLR

Version 3.0.0.9003

Date 2023-04-18

Description This package provides an FLR wrapper to the SAM state-space assessment model.

Depends R (>= 4.1.0), FLCore (>= 2.6.0), stockassessment (>= 0.12.0)

Imports methods, utils, TMB, ellipse, MASS

Suggests reshape, plyr, RColorBrewer, doParallel

Remotes fishfollower/SAM/stockassessment

Collate ``FLSAM.control.R'' ``FLSAM.R'' ``FLSAM.MSE.R'' ``FLSAM.out.R'' ``FLR2SAM.R'' ``SAM2FLR.R'' ``accessors.R'' ``af.R'' ``anf.r'' ``diagnostics.R'' ``kobe-FLSAM.R'' ``looi_retro.R'' ``methods_plots.R'' ``methods.R'' ``monteCarloStock.R'' ``pin.R'' ``zzz.R''

License GPL (>= 2)

LazyLoad no

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

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

RemoteRef HEAD

RemoteSha 4e5e7f04e363c94d206738cb4dc95f832a9ec2aa

Contents

`accessors` *Accessor functions for FLSAM*

Description

Returns results from the FLSAM asessment model

Usage

```
catchabilities(object)
power.law.exps(object)

obs.var(object)

ssb(object)
tsb(object)

n(object)
rec(object)

f(object)
fbar(object)

catch(object)
```

Arguments

`object`	An `FLSAM` or `FLSAMs` :object.

Details

Extracting many of the fitted values directly from the parameter slot of an `FLSAM` object can lead to difficulties in interpretation, due to the binding of parameters (i.e. fitting one parameter to multiple time series) and the fitting of parameters in log-space. These accessor functions provide easy access to these key results, correcting for these issues and providing both estimates of the parameter concerned and the associated confidence intervals.

The following accessor functions are currently supported:

- catchabilities - linear proportionality parameter relating the observations (e.g. survey indices) to the modelled values (e.g. numbers at age, SSB)
- power.law.exps in cases where the relationship between the observations and the modelled values is modelled as being a power-law, this function returns the associated coefficients
- obs.var - the variances estimated internally within the model that are associated with each individual time series. These variances are mathematically analogous to weighting factors
- ssb - the spawning stock biomass estimated internally within the model
- tsb - the total stock biomass estimated internally within the model
- n - the estimate number of individuals at a given age in a given year
- rec - the estimated recruitment of fish at the youngest modelled age
- f - the fishing mortality on a given age in a given year
- fbar - the mean fishing mortality in a given year
- catch - the (modelled) total catch of fish (in weight) in a given year

Accessor functions are provided for both `FLSAM` and `FLSAMs` objects.

Value

The accessor functions return a `data.frame` whoose structure is dependent on the function called. Generally the data.frame can be considered to have two parts.

The first part is the meta-data part, which gives context-specific information about the contents of each row. Possible columns returned are

- name - In cases where an FLSAMs object is supplied, the name of the corresponding object is stored in this column. There is no requirement that FLSAMs object have unique names - however, extracting and processing data can be challenging if they do not!
- fleet - The name of the fleet that the parameter corresponds to
- year - The year that the parameter corresponds to
- age - The age that the parameter corresponds to. In the case of biomass (or other non-age-based observations), age is NA.

The second part of the data-frame is common and constant to all accessor functions, and contains the actual parameters and their uncertainties

- value - The value of the parameter
- CV - The coefficient of variation for the parameter estimate i.e. standard deviation of the log-transformed parameter
- ubnd - The upper bound (95% confidence limit) for the parameter estimate
- lbnd - The lower bound (95% confidence limit) for the parameter estimate

Note

The ssb, tsb, fbar and catch accessor functions read the values estimated directly within the model. However, the estimation of these parameters is dependent on the calculation of the hessian: when the FLSAM control object has the nohess option set to true, however, the hessian will not be estimated. In these cases, these function calls will fail with a warning. If you still want to calculate the values in this case, update the corresponding stock object: see the examples.

Author(s)

Mark R. Payne

See Also

FLSAM, params, coef, coefficients

Examples

```
#Load data
library(FLSAM)
data(NSH.sam)
#Extract parameter
catchabilities(NSH.sam)
obs.var(NSH.sam)
ssb(NSH.sam)
fbar(NSH.sam)
rec(NSH.sam)
#Power law parameters could be extracted as follows, but this will
#return an error in this case, as there are no power-law exponents in the model
#power.law.exps(NSH.sam)

#And for FLSAMs
data(HERAS.sams)
ssb(HERAS.sams)
rec(HERAS.sams)
fbar(HERAS.sams)
```

AIC *Akaike's information criterion for FLSAM*

Description

Returns Akaike's information criterion for a set of FLSAM objects

Usage

```
AIC(object, ..., k = 2)
```

Arguments

object	An FLSAM or FLSAMs object
...	Further FLSAMobjects
k	the _penalty_ per parameter to be used; the default 'k = 2' is the classical AIC.

Details

The AIC criteria can be used as a basis for comparing model fits in situations where a likelihood ratio test is not appropriate (i.e. the models are not nested). For more information see the help for AIC in the stats package.

Value

Numeric value

Author(s)

Based on code by Anders Nielsen in SAM. Adapted to FLSAM by Mark R. Payne.

See Also

lr.test, AIC

Examples

```
#Load assessment
library(FLSAM)
data(NSH.sam)
#Extract AIC
AIC(NSH.sam)
#For more an FLSAMs
#data(HERAS.sams)
#plot(AIC(HERAS.sams),pch=16)
```

---

cor.plot *FLSAM sam variable correlation plot*

Description

Visualises the correlation matrix resulting from an FLSAM fit, allowing quick and easy identification of parameters that are correlated

Usage

```
cor.plot(sam, cols = c("red", "white", "blue"), full = FALSE)
```

Arguments

sam	An object of class FLSAM results from an assessment
cols	A vector of colors to form the basis for the color gradiation. The default here corresponds to be a red-white-blue scheme. Intermediate colours are formed by interpolation.
full	A logical vector indicating whether the full correlation matrix should be visualised, or just the fitted parameters i.e. without the states

Details

The stability and estimatability of a stock assessment model depends on the degree of colinearity between the parameters - situations where the parameters are co-linear or correlated can lead to instability in the model, and high sensitivity to minor changes. However, the FLSAM model has a large number of parameters that can be set, and it is relatively easy to, inadvertently, specify a configuration where the parameters are strongly correlated.

One way of identifying such problems is by visualising the correlation matrix between the parameters. This function plots the correlation coefficient (varying between -1 and 1) as a colour intensity as a function of the corresponding parameters (on the x and y axes). A strong diagonal should always be seen running through the plot - this reflects the fact that the diagonal of a correlation matrix is 1 i.e. a parameter is perfectly correlated with itself. Ideally, the remainder of the pixels (i.e. the off-diagonals) should be zero, indicating that the parameters are independent of each other. Modifications to the binding matrices may help improve the situation.

Note that the axes are labelled with the names of the parameters as stored in "params" slot. The naming of these parameters is somewhat obtuse: for more details, see the params help file. Not all parameters may be present in a fit, depending on the specific configuration of the FLSAM.control, and are therefore not plotted.

Value

An object of class "trellis". The update method can be used to update components of the object and the print method (usually called by default) will plot it on an appropriate plotting device.

Author(s)

Mark R. Payne

See Also

params, FLSAM, FLSAM.control, levelplot

Examples

```
#Load library
library(FLSAM)
data(NSH.sam)

#Generate plot
cor.plot(NSH.sam)
```

```
#Example of how to change plotting parameters
p <- cor.plot(NSH.sam)
update(p,main="Figure 1. Correlation matrix plot")
```

---

drop.from.control *Drop an element from an FLSAM.control object*

Description

Drops multiple surveys or age groups from an FLSAM.control object

Usage

```
drop.from.control(object, fleets="missing",ages="missing")
```

Arguments

object	An FLSAM.control object
fleets	A vector containing the names of the fleets to be dropped
ages	A vector containing the names of the age groups to be dropped

Details

This function can be used to modify an FLSAM.control object by removing one or more fleets, or one or more age groups from the object. Changes in the sequence of bindings that may arise due to the removal of a fleet or age group are corrected for using the update function.

Value

An FLSAM.control object

Author(s)

Mark R. Payne.

See Also

update, FLSAM.control

Examples

```
#Load assessment
library(FLSAM)
data(NSH)
#NSH.ctrl before dropping
NSH.ctrl
#Drop MLAI index
drop.from.control(NSH.ctrl,fleets="MLAI")
#Drop ages 8 and 9
drop.from.control(NSH.ctrl,ages=8:9)
```

FLSAM *FLR Wrapper for the State-space assessment model (FLSAM)*

Description

Performs a stock assessment using the State-Space assessment model (SAM)

Usage

```
FLSAM(stck, tun, ctrl, run.dir=tempdir(), batch.mode=FALSE,pin.sam=NULL)

update(object, stck, tun, run.dir=tempdir(),batch.mode=FALSE)

FLR2SAM(stck, tun, ctrl, run.dir = tempdir(),pin.sam=NULL)
runSAM(ctrl, run.dir=tempdir(),use.pin=FALSE)
SAM2FLR(ctrl="missing", admb.stem="sam", run.dir = tempdir())
```

Arguments

stck	FLStock object containing all data to be used in the assessment
tun	FLIndices object containing survey/tuning indices
ctrl	FLSAM.control object containing the parameters to be employed in the assessment
run.dir	Path to which the temporary files are to be written to or read from. Default is the R temporary directory provided by tempdir()
batch.mode	Logical variable indicating whether the code should be run in batch mode.
object	An object of the FLSAM class, resulting from e.g a call to FLSAM, that is to form the basis for the update.
pin.sam	Provide an FLSAM object, to be used as pin file, to the FLSAM function call to speed up the estimation process
use.pin	A logical variable indicating whether the SAM executable should look for a file containing initial parameter estimates (i.e. a pin file)
admb.stem	The name of the admb executable (ie without the .dat, .exe etc)

Details

FLSAM is an FLR wrapper to SAM, the state-space assessment model. SAM is written in the compiled ADMB language - compiled executables for each platform are included within the FLSAM package. The FLSAM function provides the interface between the data (FLStock, FLIndices objects) and configuration objects (FLSAM.control) stored in R, and the compiled exectable.

Three helper functions provide the core of the FLSAM method. FLR2SAM accepts the necessary data objects and writes a series of text files to the path specified by run.dir. runSAM takes care of running the SAM executable and checking that the model has run to completion correctly. SAM2FLR reads the outputs generated by the SAM model and reads them back into the FLR environment. The FLSAM function provides a wrapper around these three functions, writing the data to the temporary directory, running the executable, and reading the results back. Generally, the user should not need to use FLR2SAM, runSAM or SAM2FLR, but they can be useful in helping to debug the code.

SAM2FLR is particularly useful when trying to read results of a SAM run into FLR for plotting, further analysis etc. Ideally it should be able to figure everything out from the contents of the directory containing the files. However, if this fails, it may be necessary to give it a helping hand by creating a (FLSAM.control) object and supplying it via the "ctrl" argument.

The user can also specify the directory (via the 'run.dir' argument) in which the temporary files used to communication with SAM are stored. The default is to use the temporary directory of R.

FLSAM can also be run in "batch-mode" through the 'batch.mode' argument. This is a particularly useful feature when running a large number of assessments in an unattended situation. If the SAM model fails to converge in when running in batch-mode, it will suppress the error and return a "NULL" value (instead of an FLSAM object). If the model fails to converge when not running in batch-mode (the default), an error is returned.

It is also possible, via the 'update' function, to "reuse" the results of a previous stock assessment to provide initial guesses for the new assessment. Such a feature may be desirablie in situations where the user is running a large number of assessments that are very similar and have the same parameter configuration e.g. in a management-strategy evaluation. The supplied FLSAM object is used to generate a parameter-initialisation ("pin") file - the 'use.pin' argument in runSAM is then activated in this case, to tell the ADMB SAM executable to use these values. At the moment, it is not possible to change the assessment configuration in this manner - future releases will hopefully incorporate this feature.

Value

The typical return value is an object of the FLSAM class with the following slots.

control	The FLSAM.control object used to configure the assessment.
nohess	Logical variable indicating whether the model was fit including the calculation of the hessian
nopar	The number of parameters fitted by the model
n.states	The number of states fitted internally in the model
states	The state coupling matrix - see FLSAM.control
nlogl	The negative log-likelihood of the fitted parameters
maxgrad	The gradient of the negative log-likelihood at optimum

`vcov` The variance-covariance matrix of the fitted parameters

`params` The parameters fitted by the model, stored as a data.frame. The columns of the data frame are

- index - A numeric index for the parameter
- name - The name of parameter
- value The fitted value of the parameter
- std.dev -The standard deviation of the fitted value

`stock.n` An `FLQuant` object containing the stock-numbers at age in each year

`harvest` An `FLQuant` object containing the fishing mortality at age in each year

`residuals` A data.frame summarising the observed and fitted values, and the residuals linking them. The columns of the data frame are

- year - the year in which the observation was made
- fleet - the fleet the observation corresponds to
- age - the age the observation corresponds to. For biomass indices, this value has no meaning
- log.obs - the natural logarithm of the observed value
- log.mdl - the natural logarithm of the modelled value
- std.res - the standardised residual i.e. (log.obs - log.mdl) / obs.std, where obs.std is the observation error associated with that fleet/age.

`name` A text field giving the name of the object

`desc` A text field describing the object

`range` A named vector containg key information about range covered by the object

- min - the youngest age modelled
- max - the oldest age modelled (can be a plus group)
- plusgroup - the agegroup considered as a plusgroup (if any)
- minyear - the first year modelled
- maxyear - the last year modelled
- minfbar - the younger limit of ages used in calcualting the average fishing mortality, fbar
- maxfbar - the older limit of ages used in calcualting the average fishing mortality, fbar

`info` Contains key information about the software versions and operating system used in running the assessment

The FLSAM function will return an FLSAM object in all cases when the model converges. If the model does not converge, FLSAM will return an error unless it is running in batch mode, in which case it will return "NULL".

runSAM() returns an integer error code corresponding to the return value of the SAM model - 0 for a successful run, non-zero otherwise. SAM2FLR will return an FLSAM object. FLR2SAM does not return any values.

Author(s)

SAM2FLR based on code by Anders Nielsen and adapted by Mark Payne. FLR2SAM and FLSAM by Mark Payne

References

Nielsen et al (2012) In prep.

See Also

FLSAM.control

Examples

```
#Load library
library(FLSAM)
#Prepare data objects
data(NSH)
data(NSH.sam)
#Perform assessment
#Takes about two minutes on a 2.4GHz workstation
sam <- FLSAM(NSH,NSH.tun,NSH.ctrl)
sam <- FLSAM(NSH,NSH.tun,NSH.ctrl, return.fit=TRUE)

#If you want to re-use the sam object to create a pin file, use this:
sam <- FLSAM(NSH,NSH.tun,NSH.ctrl,pin.sam=sam)
#View results
plot(sam)

#Repeat the assessment using the previous result as an initial guess
uninit.time <- system.time(sam.uninit <- FLSAM(NSH,NSH.tun,NSH.ctrl))
#init.time   <- system.time(sam.init   <- update(sam.uninit,NSH,NSH.tun,NSH.ctrl))
#Big speed increase! (your results may vary: ours are about 2x)
#uninit.time[3]/init.time[3]
#But assessments are the sam (phew!)
#plot(FLSAMs(uninitialised=sam.uninit,initialised=sam.init))
```

FLSAM.control *FLSAM.control*

Description

Creates an FLSAM.control that can be used to control an FLSAM assessment

Usage

```
FLSAM.control(stck, tun, default="full",sam.binary="missing")
```

Arguments

stck	An FLStock object
tun	An FLIndices object

default	Specifies how the control object should be populated. Default option is "full", which takes a reasonable first guess at a configuration. Any other value will populate all matrices with NA values.
sam.binary	Path and name of the SAM executable file to run

Details

An FLSAM assessment requires three elements - an FLStock object, an FLIndices object and an FLSAM.control object. The FLSAM.control object contains the configuration options for the model, and is therefore key to the process. However, there are a great many configuration options and therefore setting up the control object is a relatively complex process. The purpose of the FLSAM.control function is to create a template (possibly containing sensible first guesses at a configuration) object that can then be populated by the user according to their wishes.

The key concept with the configuration of a SAM model is that some parameters are bound together, so that the model fits just a single parameter, but uses it in multiple places e.g. by using a single parameter for the catchabilites of ages 5-9 in a survey. We refer to this concept as "binding" the parameters together.

Bindings are expressed in a matrix format in the FLSAM.control object. The rows of the matrix correspond to the fleets, whilst the column correspond to the ages - it is important to remember that while all age-fleet combinations are represented here, not all are valid i.e. a fleet may not contain information on all ages. Bindings are specified with integers - the integers indicate which parameter should be used in the specific fleet-age context represented by that point in the matrix. Repeating a number means that all those fleet-age combinations with the same number will be fitted with just one parameter i.e. bound together. Parameters that are not to be fitted, or are turned off, are specified by setting the value to NA.

FLSAM.control can populate the template in two different ways, as indicated by the "default" argument. If default="full", a reasonable first guess at a configuration is supplied - this consists of linear catchability models for the surveys and a fixed selectivity pattern in the fishery. Alternatively, if "default" is set to any other value, all values are set to NA and therefore turned "off".

Earlier versions of FLSAM contained the executable (binary) file within the package distribution. However, this version is designed to interface with customised versions of the sam model, which can be modified and compiled locally. The sam.binary slot in the FLSAM.control object can be used to specify the location of this executable file. If the file is not specified, FLSAM reverts to the versions stored within the package - however, in the future, these binaries may be removed.

Value

Returns an FLSAM.control object with the following slots

name	A text field giving the name of the object
desc	A text field describing the object
range	A named vector containg key information about range covered by the object • min - the youngest age modelled • max - the oldest age modelled (can be a plus group) • plusgroup - the agegroup considered as a plusgroup (if any) • minyear - the first year modelled

- maxyear - the last year modelled
- minfbar - the younger limit of ages used in calcualting the average fishing mortality, fbar
- maxfbar - the older limit of ages used in calcualting the average fishing mortality, fbar

`fleets`	A named vector indicating the type of data that it represents. 0 = catch. 1=<I don't know! "con" something>. 2=Numbers at age survey. 3. SSB index
`plus.group`	Logical specifying whether the oldest group should be considered as a plus group
`states`	Binds the fishing mortality random-walks together
`logN.vars`	Binds the variances of the numbers-at-age random-walks together
`catchabilties`	Binds the catchabilities of the surveys together
`power.law.exps`	Allows power-law catchability models to be implemented, and possibly bound together. In the case where a value is set to NA, a linear model will be employed for that fleet-age combination
`f.vars`	Binds the variances of the fishing mortality random-walks together (as opposed to "states", which binds the value of the random walks i.e. to bind one random walk to multiple fishing-mortalties at age).
`obs.vars`	Binds the variances (standard deviations) of the observations together
`cor.F`	Logical variable indicating whether the fishing mortality random walks in a manner that incorporates correlations between the ages. Default is FALSE (i.e. the fishing mortalities are completely independent).
`srr`	Stock recruitment relationships. Value options are 0= Random walk. 1= Ricker. 2=Beverton-Holt.
`nohess`	Logical variable indicating whether the hessian of the fitted model should be estimated. Default is TRUE. Turning on this variable can be used to increase the speed of the model but it does so at the cost of removing all uncertainty estimates and the variance-covariance matrix from the fitted object. Any methods that are dependent on these values will most likely fail as a result.
`timeout`	Model timeout setting (in seconds). If the model fails to finish within this time-frame, it exits and returns and error. Default is 3600s (one hour).
`sam.binary`	Name and path of the executable file to be run

Warning : It is important to realise that the default configuration returned by FLSAM.control is almost certaintly not the best configuration of your model - it is merely a first guess setup for your convenience. Similarly, there is no guarantee that this model will lead to a converged solution. It is the users responsibility to find the most appropriate "best" configuration of the model.

Author(s)

Mark R. Payne and Niels Hintzen

See Also

FLSAM

Examples

```
#Load data
library(FLSAM)
data(NSH)
#Create an object
ctrl <- FLSAM.control(NSH,NSH.tun)
ctrl
#Bind HERAS catchabilities on ages 5:8
ctrl@catchabilities["HERAS",as.character(5:8)] <- 11
#Alternatively can also use update functionality
ctrl@catchabilities["HERAS",as.character(5:8)] <- 101
ctrl <- update(ctrl)
#See example of full control object
data(NSH.sam)
NSH.ctrl
```

FLSAM.out *Create FLSAM formatted output file*

Description

Document the results of the FLSAM assessment including all input values, model settings and output in formatted tables

Usage

```
FLSAM.out(stck, tun, sam, format = "TABLE %i.")

sam.out(stck, tun, sam, format = "TABLE %i.")
```

Arguments

stck	An FLStock object to be documented
tun	An FLIndices object to be documented
sam	An FLSAM assessment model output object
format	The Title to be specified with each table.

Details

This function produces a set of tables that can be used to directly document an assessment. Included in the tables are all (relevant) input data, the details of the tuning indices, the model parameter configuration, the estimated parameters and all estimated values. The structure and approach is inspired by the "ica.out" documentation from the ICA package.

sam.out is simply an alias for FLSAM.out.

Value

The output is a table that can be written to disk and pasted directly into any text editor.

Author(s)

N.T. Hintzen

Examples

```
data(NSH)
data(NSH.sam)

sam.out <- FLSAM.out(NSH,NSH.tun,NSH.sam,format="Table %i")

#-Set margins to make prettier output
old.opt           <- options("width","scipen")
options("width"=80,"scipen"=1000)

#Write the file to the working directory
write(sam.out,file="sam.out")
#-Reset margins
options("width"=old.opt$width,"scipen"=old.opt$scipen)
```

FLSAMs-class *Class* "FLSAMs"

Description

FLSAMs is a class that stores multiple FLSAM objects in a single container, allowing quick and easy processing and plotting.

Objects from the Class

Objects can be created by calls of the form FLSAMs(...).

Slots

.Data: The data "list" ~~

names: Names of the elements "character" ~~

desc: Description of the object "character" ~~

lock: Lock mechanism. If turned on. the length of the list cannot be modified by adding or removing elements "logical" ~~

Extends

Class "FLlst", list vector

Methods

+ signature(e1 = "FLSAMs", e2 = "FLStock"): ...

+ signature(e1 = "FLStock", e2 = "FLSAMs"): ...

catchabilities signature(object = "FLSAMs"): ...

coefficients signature(object = "FLSAMs"): ...

coef signature(object = "FLSAMs"): ...

f signature(object = "FLSAMs"): ...

FLSAMs signature(object = "FLSAMs"): ...

lr.test signature(object = "FLSAMs"): ...

obs.var signature(object = "FLSAMs"): ...

power.law.exps signature(object = "FLSAMs"): ...

Author(s)

Mark R. Payne and Niels Hintzen

See Also

accessors, HERAS.sams

Examples

```
data(NSH)
data(NSH.sam)

#Modify HERAS catchability bindings systematically
#Note that this is the same code used to create the
#HERAS.sams data object
ctrl.list <- list(all.free=3:10,
                  four.lvls=c(3,3,4,4,5,5,6,6),
                  three.lvls=c(3,3,4,4,5,5,5,5),
                  two.lvls=c(3,3,3,4,4,4,4,4))

ctrls <- lapply(ctrl.list,
  function(x) {
    tmp.ctrl <- NSH.ctrl
    tmp.ctrl@catchabilities["HERAS",ac(1:8)] <- x
    return(update(tmp.ctrl))
})

sam.list <- lapply(ctrls,FLSAM,stck=NSH,tun=NSH.tun)
HERAS.sams <- FLSAMs(sam.list)

#Plot results
plot(HERAS.sams)

#Plot catchability coefficients for different levels
qs.dat <- catchabilities(HERAS.sams)
```

```
xyplot(value ~ age,data=qs.dat,groups=name,type="b",
    subset=fleet=="HERAS",auto.key=TRUE)

#Update stock object
NSH.HERAS <- NSH + HERAS.sams
plot(NSH.HERAS)

#Compare models with AIC
HERAS.aics <-AIC(HERAS.sams)
plot(HERAS.aics,xaxt="n",xlab="Model",ylab="AIC")
axis(1,at=seq(HERAS.aics),labels=names(HERAS.aics)) #three lvls looks best
```

---

looi *Run a sequence of assessments leaving one fleet (non-catch) in or out*

Description

Perform a sequence of assessments where each of the fleets are excluded or included from the assessment run. It is also possible to run intermediate states where 2 or more fleets are excluded or included from the assessment

Usage

```
looi(stck, tun, ctrl, type="full")

loo(stck, tun, ctrl)
loi(stck, tun, ctrl)
```

Arguments

stck	FLSAM object that specifies the stock characteristics
tun	FLIndices object to use within the 'looi' assessment and out of which fleets will be included or dropped
ctrl	FLSAM.control object that specifies the assessment settings
type	Selects the type of LOOI run to perform. Options are • type: 'full' runs all combinations of leaving fleets in and out (may take a long time to run) • type: 'LOI' runs all combinations keeping only 1 fleet in • type: 'LOO' runs all combinations leaving only 1 fleet out. Default value is 'full'.

Details

The 'loo' and 'loi' methods call the 'looi' method with the appropriate type selected, and can be used as convenient shotcut

Value

An FLSAMs object is returned where the names slot contains the combination of fleets used within the assessments

Author(s)

Niels T. Hintzen and Mark R. Payne

See Also

FLSAMs

Examples

```
#- Load data
data(NSH)
data(NSH.sam)

#- Run a LOI
one.in <- loi(NSH,NSH.tun,NSH.ctrl)
plot(one.in)

#- Run a LOO
one.out <- loo(NSH,NSH.tun,NSH.ctrl)
plot(one.out)
```

---

lr.test *Perform a log-ratio test for all models specified*

Description

Compare model fit performance among the models specified based on log-likelihood ratio's. Different in log-likelihood is calculated and degrees of freedom are given. P values are specified assuming a chi-quared distribution.

Usage

```
lr.test(object,...,type="sequential")
```

Arguments

object	An FLSAM or FLSAMs object
...	In combination with an FLSAM object: additional objects of the same type (need at least 1)
type	The type of comparison to make. A "sequential" comparison compares object 1 against object 2, object 2 against object 3,... whilt a "first" comparison compares object 1 against object 2, object 1 against object 3,...

Value

A summary table of the test characteristics is returned

Warning!

Log-likelihood tests can only be used for nested models (i.e. model parameterisation is different) and are not appropriate for different input data or different model types.

Author(s)

Code by Anders Nielsen, implemented by Niels T. Hintzen and Mark R. Payne

See Also

looi

Examples

```
#- Load the data
data(HERAS.sams)

#- Run lr ratio test on example FLSAMs object
lr.test(HERAS.sams)

#- Run lr ratio test on individual FLSAM objects
lr.test(HERAS.sams[[1]],HERAS.sams[[2]],HERAS.sams[[3]])
```

---

monteCarloStock *Generate new starting conditions of numbers and f-at-age.*

Description

Method to sample from the variance-co-variance matrix and based on a multi-variante normal distribution generate new starting conditions of numbers-at-age and f-at-age.

Usage

```
monteCarloStock(stck,sam,realisations,run.dir=tempdir())
```

Arguments

stck	An object of class FLStock
sam	An object of class FLSAM
realisations	Number of newly generated starting conditions.
run.dir	Path to which the temporary files are to be written to or read from. Default is the R temporary directory provided by tempdir()

Details

Large number of realisations might take longer to create.

Value

An FLStock with iter=number of realisations is returned

Author(s)

N.T. Hintzen

See Also

FLSAM

Examples

```
data(NSH)
data(NSH.sam)

#Stock with one iter
summary(NSH)#iter = 1
plot(NSH)
#Generate new stock with 100 iters
#newStock <- monteCarloStock(NSH,NSH.tun, NSH.sam,100, ncores=2L)
#Now does it look?
#summary(newStock)#iter = 100
#plot(newStock)  #Voila!
```

NSH *North Sea Autumn Spawning Herring Stock Assessment*

Description

A set of FLR objects used to perform the 2012 North Sea Autumn Spawning Herring Stock Assessment

Usage

```
data(NSH)
data(NSH.sam)
data(HERAS.sams)
```

Details

The data is split into three separate groups. The first of these, returned by `data(NSH)` contains objects relating to the North Sea herring stock

- NSH - `FLStock` object for the North Sea herring stock, containing data relating to the stock e.g. catch numbers at age
- NSH.tun - `FLIndices` object containing the results of surveys on the North Sea herring stock. The four surveys are
  - MLAI - The Multiplicative Larval Abundance Index. Used as a index of SSB
  - IBTS0 - A index of late-larval abundance performed in February. Used as an index of age 0 fish
  - IBTS-Q1 - The International Bottom Trawl Survey, performed in February. Used as an index of numbers-at-age for age 1-5 fish
  - HERAS - The Herring Acoustic survey, performed in July. Used as an index of numbers at age for age 1-9 fish.

In addition the results of an assessment of this stock performed using FLSAM, and the associated configuration file, are returned by `data(NSH.sam)`

- NSH.sam - An `FLSAM` object containg the results of the assessment
- NSH.ctrl - An `FLSAM.control` object containg the configuration used to perform the assessment

The third group is an example of an `FLSAMs` object and is called `HERAS.sams`. It contains four `FLSAM` objects resulting from various parameter configurations for the HERAS survey, grouping various age groups together. The four objects correspond to all age groups having independent catchability parameters ("all-free"), catchability parameters for ages 1-2, 3-4, and 5-8 ("three.lvls"), for ages 1-3, and 4-8 ("two.lvls") and one catchability parameter for all ages ("one.lvl").

Note that these are not the full North Sea herring assessment objects, but rather have been truncated to remove the data prior to 1992, so as to allow the model to converge quickly. The full assessment can be found in the Herring Working Group repository, "http://hawg.googlecode.com".

Source

ICES. 2012. Report of the Herring Assessment Working Group for the Area South of 62 N (HAWG), 13 - 22 March 2012, Copenhagen, Denmark. ICES CM 2012/ACOM:06. 835 pp.

Examples

```
#Load data
data(NSH)
data(NSH.sam)
#Stock summary plot
plot(NSH.sam)
#Update NSH FLStock object
NSH <- NSH + NSH.sam
plot(NSH)
#Example of FLSAMs object
data(HERAS.sams)
```

```
#Likelihood ratio test to choose best combination
lr.test(HERAS.sams)
```

---

obscv.plot *Create diagnostic observation variance v.s. CV plot.*

Description

Plot the estimated observation variance of each data source in the assessment agains the CV of the estimate.

Usage

```
obscv.plot(sam)
```

Arguments

sam	An FLSAM assessment output object

Details

This plot can be used as rough and ready way to gauge how "tightly" the SAM model has converged on the observation variances. In cases where the observation variances are poorly defined, the associated CV of the estimate will be large. In such cases, the model convergence should be examined in more detail.

Value

A plot is returned with on the horizontal axis the observation variance of each data source versus on the vertical axis the CV estimate associated with the data source

Author(s)

Mark Payne, Niels T. Hintzen

See Also

obsvar.plot,residual.diagnostics

Examples

```
data(NSH.sam)

obscv.plot(NSH.sam)
```

`obsvar.plot` *Create diagnostic observation variance plot.*

Description

Plot the estimated observation variance of each data source in the assessment as a sorted barplot.

Usage

```
obsvar.plot(sam)
```

Arguments

`sam`	An FLSAM assessment output object

Details

Observation variances in the SAM model are analogous to the "weightings" employed in other model - in this case, they represent how noisy each of the individual tuning fleets are. Close examination of the values of these observation variances can given valuable insight into what the model is doing, and where it thinks the most information is.

Value

A barplot is returned with fleets at the horizontal and observation variance at the vertical axis.

Author(s)

Mark Payne, Niels T. Hintzen

See Also

`obscv.plot`,`residual.diagnostics`

Examples

```
data(NSH.sam)

obsvar.plot(NSH.sam)
```

`otolith` *Visualise uncertainity in the terminal Fbar and SSB*

Description

Generates and visualises the two-dimensional confidence interval from a stock assessment based on a parametric bootstrapping exercise.

Usage

```
otolith(object, year=sam@range["maxyear"], plot=TRUE, show.points=FALSE, do.contours=TRUE, margin.p
```

Arguments

`object`	An FLSAM object upon which the stock assessment is based.
`year`	Year the otolith is based upon.
`plot`	Boolean. Whether the plot should be generated or whether or not.
`show.points`	Boolean. Show the individual point generated by the parametric bootstrap.
`do.contours`	Boolean. Generate contours showing the confidence intervals in the two dimensions.
`margin.plots`	Boolean. Generate horizontal and vertical plots to the sides of the main plot showing the 1-D confidence intervals.
`show.estimate`	Boolean. Show the point estimate generated by the assessment.
`xlim, ylim`	Two element numeric vectors giving the axis limits for the x- and y- axes respectively.
`n`	The number of individual parametric bootstrap values to generate.
`pch`	The symbol used to plot the individual points. See `points` for more information.
`alpha`	The confidence interval plotted in the margin plots. Defaults to 0.05 (95%).
`show.grid`	Boolean. Display a grid in the background.
`n.grid`	The density of the grid used to generate the two-dimensional uncertainty surface which the contours characterise. The default value of 50 works well for most applications, although it can be increased if the number of bootstrap estimates is also increased.
`contour.args`	A list of arguments suitable for use in the `contour` function controlling the formatting of the contour lines.
`...`	Other arguments suitable to be feed to the standard `plot` function.

Details

The function might become very slow when a large number of n is required while the variance-covariance matrix is large.

Author(s)

Mark Payne & Susan Lusseau

See Also

sprintf, options

Examples

```
#Load assessment objects
data(NSH.sam)
otolith(NSH.sam,year=2010,n=150)
otolith(NSH.sam,year=2010,n=1e4)

otolith(NSH.sam,year=2010,n=1e4,show.points=TRUE)
```

---

parameter.accessors *Parameter accessor functions for FLSAM*

Description

Returns parameters fitted by the FLSAM asessment model

Usage

```
params(object,...)
coef(object,...)
coefficients(object,...)
```

Arguments

object	An FLSAM or FLSAMs object.
...	Other arguments

Details

These accessor functions provide direct access to the values returned by the SAM model. However, interpretation of these parameters can be challenging and requires some knowledge of the underlying workings of the SAM model. For more access in a more user-friendly format, see the accessors help.

The following parameter accessor functions are currently supported:

- params - returns all values estimated by the model, including derived parameters such as SSB, TSB, etc.
- coef - returns the parameters fitted by the model, but not derived parameters such as SSB, TSB etc.

- coefficients - an alias for coef

Accessor functions are provided for both FLSAM and FLSAMs objects.

Note that the naming of these parameters is somewhat obtuse. A brief description of these parameters follows:

- logFpar: Log-transformed catchability of the numbers-at-age survey indices. The binding of these parameters are controlled in the 'catchabilities' slot of the FLSAM.control object.
- logSdLogFsta: Log-transformed standard deviation of the random walk in the (log-transformed) fishering mortalities. Bindings are controlled in the 'f.vars' slot.
- logSdLogN: Log-transformed standard deviation of the random walk in (log-transformed) N. Bindings are controlled in the logN.vars slot
- logSdLogObs: Log-transformed observation variance for each data source. Bindings are controlled in the 'obs.vars' slot
- logScaleSSB: Log-transformed catchability of the SSB indices
- logSdSSB: Log-transformed observation variance of the SSB indices
- rec_loga: Log-transformed 'a' parameter of the stock-recruitment model. Fitting of this parameter is controlled by the 'srr' slot
- rec_logb: Log-transformed 'b' parameter of the stock-recruitment model. Fitting of this parameter is controlled by the 'srr' slot
- rho: Correlation coefficient in the correlated random-walk in fishing mortality. Fitting of this parameter is controlled by the 'cor.F' slot
- logPowSSB: Log-transformed power-law coefficient of the SSB catchability model.
- U: The state variables. The order of the state variables is N(year 1),F(year 1, N(year 2), F(year 2), where N are the numbers-at-age state variables and F the fishing mortality state variables)

Not all parameters may be present in a fit, depending on the specific configuration of the FLSAM.control.

Value

The parameter accessor functions return a data.frame whoose length depends on the calling function. params() returns all values (both fitted and dervied parameters), whereas coef() and coefficients() only return the fitted parameters.

Columns returned are

- object.name - In cases where an FLSAMs object is supplied, the name of the corresponding object is stored in this column. There is no requirement that FLSAMs object have unique names - however, extracting and processing data can be challenging if they do not!
- index - An indexing number produced internally by SAM
- name - The name of the parameter
- value - The value of the parameter
- std.dev - The standard deviation of the fitted parameter

Author(s)

Mark R. Payne

See Also

FLSAM, accessors

Examples

```
#Load data
library(FLSAM)
data(NSH.sam)
#Extract parameter
catchabilities(NSH.sam)
obs.var(NSH.sam)
ssb(NSH.sam)
fbar(NSH.sam)
rec(NSH.sam)
#Power law parameters could be extracted as follows, but this will
#return an error in this case, as there are no power-law exponents in the model
#power.law.exps(NSH.sam)

#And for FLSAMs
data(HERAS.sams)
ssb(HERAS.sams)
rec(HERAS.sams)
fbar(HERAS.sams)
```

---

procerr.plot *Produce diagnostics on process error in the SAM Model*

Description

Generate plots that inform on process error in the SAM assessment model

Usage

```
procerr.plot(stck, weight = "stock.wt", type = "n", rel = F)
```

Arguments

stck	An FLStock object
weight	Option to choose between stock.wt and catch.wt
type	Option to choose between observation error related to N-at-age ('n'), mortality ('mort') or total biomass ('tsb')
rel	Logical. Express error relative or absolute

Value

A picture is returned

Author(s)

Niels T. HIntzen (based on code from Einar Hjorleifsson)

Examples

```
library(FLSAM)

data(NSH)

procerr.plot(NSH,type="n")
procerr.plot(NSH,type="n",rel=TRUE)
procerr.plot(NSH,type="mort")
procerr.plot(NSH,type="mort",rel=TRUE)
procerr.plot(NSH,type="tsb",weight="catch.wt",rel=FALSE)
procerr.plot(NSH,type="tsb",weight="catch.wt",rel=TRUE)
procerr.plot(NSH,type="tsb",weight="stock.wt",rel=FALSE)
procerr.plot(NSH,type="tsb",weight="stock.wt",rel=TRUE)
```

---

residual.diagnostics *FLSAM assessment diagnostics*

Description

Generate a set of assessment diagnostics from an FLSAM object.

Usage

```
residual.diagnostics(x,title=x@name)
```

Arguments

x	An FLSAM object resulting from a stock assessment.
title	A text string to be displayed as part of the plot titles. Defaults to the name of the FLSAM object.

Details

The SAM model assumes that the residuals between the observed and fitted values are log-normally distributed, and independently and identically distributed (i.i.d.). This function produces a set of diagnostic plots allowing these assumptions to be validated, in much the same way as they would be done in a linear modelling situation. The function produces a set of plots (six in all) for each individual combination of age and fleet (including both catches and surveys). The six panels produced are

1. Time series of fitted and observed values. A direct comparison of the individual time series - fitted values are linked by a line, while observations are plotted as points. In an ideal fit, the observations should closely agree with the fitted values.

2. Observed vs fitted values. The fitted values are plotted on the horizontal axis and the obseved values on the vertical axis. A 1:1 line, indicating a perfect fit between the two, is also plotted for reference. In an ideal fit, the points will be distributed tightly around the 1:1 line, with no or few outliers.
3. Residuals over time. The residuals are plotted as a time series, with a line connecting the point to the zero line. In an ideal fit, the points should be distributed randomly about the zero line, with no systematic patterns.
4. Tukey-Anscombe plot. Residuals are plotted as a function of the fitted values. A loess smoother (red line) is also plotted. In an ideal fit, the residuals should be evenly distributed about zero with constant mean and variance - the red smoother line can aid in detecting systematic changes in the residuals.
5. Normal QQ-Plot. The residuals are plotted against the corresponding quantiles of the normal distribution. A 1:1 line is plotted together with the qqline (passing through the first and third quantiles - see help for qqline()). In an ideal fit, the qqplot will show a straight line, especially in the tails, indicating that the residuals are normally distributed.
6. Autocorrelation plot. Plots the autocorrelation between residuals as a function of the lag, with confidence intervals for the significance of the correlation. In an ideal fit, the autocorrelations should be non-significant, particularly at the shortest time lags.

Residuals in SAM are standardised log-residuals (also known as studentized residuals). They are defined as the residual (log (observed value) - log(fitted value)) divided by the standard deviation of the observations (the observation error).

Both observed and fitted values are plotted on a logarithmic scale - FLSAM fits all observations as being log-transformed (i.e. a log-normal error structure) and therefore plotting on a logarithmic scale is the most appropriate. Furthermore, the top four plots are laid-out so that all scales are the same, and that individual points can be traced between the plots by moving horizontally or vertically as appropriate.

All plots are plotted sequentially and immediately, one after the other. If this is a problem (e.g. for plotting immediately to a window), a pause can be inserted between each page using par(ask=TRUE) - see the examples.

Value

None.

Author(s)

Based on ideas and code developed by Hans Bogaard. Further refinements by Susan Clarke and Mark R. Payne. This implementation by Susan Lusseau.

See Also

`qqplot` and `qqline` for information on QQ-plots. `plot.acf` and `acf` for information on the autocorrelation function plot

Examples

```
#Load data
library(FLSAM)
data(NSH.sam)
#Plot each page with a pause between
par(ask=TRUE)
#Generate plots
residual.diagnostics(NSH.sam)
```

`retro` *Retrospective stock assessment for FLSAM*

Description

Performs a retrospective stock assessment for the desired years using the FLSAM stock assessment method

Usage

```
retro(stock, indices, control, retro, year.range)
```

Arguments

`stock`	An `FLSAM` object
`indices`	An `FLIndices` object
`control`	An `FLSAM.control` object
`retro`	An integer that specifies the number of retrospective years. Default value = 0. Only used if year.range is not specified.
`year.range`	Numeric vector of years to perform the assessment

Details

The argument 'year.range' is a numeric vector of years for which the assessment is to be performed. If this is not specified the integer "retro" used (default value = 0). If retro = 0 the assessment is run for final year only. If retro = 1, the assessment is run for the penultimate and final year and so on.

Value

Returns an FLSAMs object. Individual assessments that failed to converge are not included in this object, but a warning is issued in each case.

Author(s)

Based on code by Laurence Kell, modified by Niels T. Hintzen and Mark R. Payne

See Also

looi

Examples

```
#Load assessment
library(FLSAM)
data(NSH)
data(NSH.sam)

#Run a retrospective analyses for 3 years
res <- retro(NSH,NSH.tun,NSH.ctrl,retro=3)

#Make an FLStocks object and plot the results
NSH.retro <- res + NSH
plot(NSH.retro)
```

Index