Package: bbm (via r-universe)

September 7, 2024

Title FLR Implementation of a Two-Stage Stock Assessment Model

Version 0.0.3

Description The two-stage biomass-based model for the Bay of Biscay anchovy (Ibaibarriaga et al., 2008).

Depends R(>= 3.4.0), methods, FLCore, ggplotFL

Imports TMB, stats

LinkingTo TMB, RcppEigen

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

Suggests testthat, knitr, rmarkdown, latticeExtra

VignetteBuilder knitr

License EUPL

LazyLoad Yes

LazyData No

Roxygen list(markdown = TRUE)

BugReports https://fishreg.jrc.ec.europa.eu/gitlab/mosquia/bbm/issues

Collate ``generic_methods.R'' ``bbmControl_class.R'' ``bbmFLPar_fun.R'' ``bbmFit_class.R'' ``bbmFitresiduals_class.R'' ``bbm_method.R'' ``calcPop_fun.R'' ``calcIndices_fun.R'' ``createInits_method.R'' ``simIndices_method.R'' ``periods_fun.R'' ``functions_auxiliary.R'' ``BoBane_data.R''

Encoding UTF-8

RoxygenNote 7.2.1

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

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

RemoteRef HEAD

RemoteSha c449fad0f3101a663cb5a0e63dcab8d8fe8ebb7c

Contents

---

bbm *Method bbm*

Description

Function to fit a two-stage biomass-based model.

Usage

```
bbm(object, indicesB, indicesP, ...)

## S4 method for signature 'FLQuant,FLQuants,FLQuants'
bbm(object, indicesB, indicesP, findicesB, findicesP, control, inits)

## S4 method for signature 'FLStock,ANY,ANY'
bbm(
  object,
  indicesB,
  indicesP,
  findicesB = NULL,
  findicesP = NULL,
  control,
  inits
)

## S4 method for signature 'FLQuant,FLIndices,FLIndices'
bbm(
  object,
  indicesB,
  indicesP,
```

```
  findicesB = NULL,
  findicesP = NULL,
  control,
  inits
)

## S4 method for signature 'FLQuant,FLQuant,FLQuant'
bbm(object, indicesB, indicesP, findicesB, findicesP, control, inits)
```

Arguments

object	An FLQuant with catch information (in mass) or an FLStock. These objects must only have two age classes (recruits and adults) and the number of seasons should be 1 or the number of seasons determined by the timing of the different indices.
indicesB	Abundance indices in total biomass (element of class: FLQuants, FLQuant or FLIndices) from surveys. Please assign a survey name to each index.
indicesP	Percentage of recruits in biomass (element of class: FLQuants, FLQuant or FLIndices) from surveys. Please assign a survey name to each proportion.
findicesB	A vector with fraction of the year corresponding to each of the indicesB.
findicesP	A vector with fraction of the year corresponding to each of the indicesP.
control	A bbmControl with control arguments.
inits	An FLPar with initial values.

Value

An object of class bbmFit.

Author(s)

Leire Ibaibarriaga & Sonia Sanchez.

See Also

bbmFit, FLQuant, FLQuants, bbmControl, FLPar, bbmFLPar

Examples

```
# Load data
data(ane)
# Case:  object='FLStock'; indicesB=indicesP='FLIndices'; control='bbmControl'; inits='FLPar'
stock <- FLStock(catch.n=catch.ane, catch.wt=catch.ane*0+1)
units(stock@catch.wt) <- ''
stock@catch <- quantSums(stock@catch.n*stock@catch.wt)
run2 <- bbm(stock, indicesB=indicesB.ane, indicesP=indicesP.ane, control=control.ane, inits=inits.ane)
# Case:  object='FLQuant'; indicesB=indicesP='FLIndices'; control='bbmControl'; inits='FLPar'
run3 <- bbm(catch.ane, indicesB=indicesB.ane, indicesP=indicesP.ane, control=control.ane, inits=inits.ane)
# Case:  object='FLQuant'; indicesB=indicesP='FLQuant'; control='bbmControl'; inits='FLPar'
namdel <- c("q_acoustic","psi_acoustic","xi_acoustic") # we will take only one of the indices --> need to delete the
```

```
control <- control.ane
control@param.fix <- control@param.fix[dimnames(control@param.fix)$params[!dimnames(control@param.fix)$params %
inits  <- inits.ane[dimnames(inits.ane)$params[!dimnames(inits.ane)$params %in% namdel],]
run4 <- bbm( catch.ane, indicesB=indicesB.ane[[1]]@index, indicesP=indicesP.ane[[1]]@index,
         findicesB=c( depm=(indicesB.ane[[1]]@range[['startf']]+indicesB.ane[[1]]@range[['endf']])/2),
         findicesP=c( depm=(indicesP.ane[[1]]@range[['startf']]+indicesP.ane[[1]]@range[['endf']])/2),
             control=control, inits=inits)
# Plot assessed populations
runs <- list(run2=run2, run3=run3, run4=run4)
biomass <- FLQuants(lapply(lapply(runs, stock.bio), quantSums))
plot(biomass)
```

---

bbmControl *bbmControl class*

Description

bbmControl class is used for setting the control parameters for the bbm function.

Usage

```
bbmControl(object, ...)

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

Slots

g Instantaneous rate of biomass decrease, g = M - G, for recruits (rec) and adults (adult), c(rec=numeric(1), adult=numeric(1)).

param.fix An FLPar with value 1 if the parameter is fixed at its initial value and 0 otherwise.

Validity

Dimensions g must be a vector of length 2 and with the names 'rec' and 'adult'

Class param.fix must be of class FLPar

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

Accessors

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

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.

Author(s)

Leire Ibaibarriaga & Sonia Sanchez

See Also

bbm

Examples

```
# Load data
data(ane)

# Generate an object of FLPar class (different alternatives)
bbmControl()                    # empty object
slotNames(bbmControl())         # slots

bbmControl( g=c(rec=0.68, adult=0.68), param.fix=FLPar(nyear=20, nindex=3)) # setting values for the slots

# Run assessment (control must be of class bbmControl)
class(control.ane)
run <- bbm(catch.ane, indicesB=indicesB.ane, indicesP=indicesP.ane, control=control.ane, inits=inits.ane)
run
```

---

bbmFit *bbmFit class*

Description

bbmFit class is used for storing the output of the bbm function. This includes abundance estimates in biomass (for recruits and adults) and information on the model fit.

Usage

```
bbmFit(object, ...)

## S4 method for signature 'missing'
bbmFit(
  object,
  years = "missing",
  niter = "missing",
  namesB = "missing",
  namesP = "missing",
  ...
)

## S4 method for signature 'FLStock,bbmFit'
e1 + e2
```

```
## S4 method for signature 'bbmFit'
residuals(object)

## S4 method for signature 'bbmFit'
logLik(object, ...)

## S4 method for signature 'bbmFit,ANY'
AIC(object, ..., k = 2)

## S4 method for signature 'bbmFit'
BIC(object, ...)

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

Arguments

obj	The object to be subset
it	Iteration(s) to be extracted

Slots

input Input data. list, Containing the following information: catch, indicesB, indicesP, perindicesB, perindicesP, control, f and nper.

convergence Convergence code, vector(niter). Where 0 indicates successful completion. For other possible error codes see ?optim.

message Character string giving any additional information returned by the optimizer, or "".

fitSumm Fit summary (with information on 'nlogL', 'nobs', 'nopar'), array[3,niter].

params Estimated parameters in bbm function, FLPar[npar,niter], in linear scale.

params.se Standard errors in parameters' estimates. FLPar[npar,niter], in linear scale.

vcov Variance-covariance matrix, array[npar,npar,niter].

stock.bio Estimated stock biomass for recruits and adults in the different seasons, where seasons are dertermined by the index times. FLQuant with two age classes: recruits and adults.

indicesB Estimates of surveys' total abundances in biomass, FLQuants.

indicesP Estimates of surveys' percentage of recruits in biomass, FLQuants.

Validity

Dimensions
- age: stock.bio must be an FLQuant with only 2 age classes (recruits and adults) and each index in indicesB must be an FLQuant with only 1 age class ('all')
- year, unit, season, area: equal for stock.bio, indicesB and indicesP
- iter: equal for stock.bio, convergence, fitSumm, indicesB and indicesP

Parameters Same number of parameters required in params, params.se and vcov

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

Accessors

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

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 `years`, `niter`, `namesB` or `namesP` are provided, this is used for sizing and naming the different slots.

Methods

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

+ Updates an `FLStock` with new information on the BBM assessment.

residuals Calculates Pearson residuals, returns an object of class `bbmFitresiduals`.

logLik Method to extract Log-Likelihood, returns an object of class `logLik`.

AIC Method to calculate Akaike's 'An Information Criterion' (AIC) of a `bbmFit` object from the value of the obtained log-likelihood stored in its `logLik` slot.

BIC Method to calculate the Bayesian information criterion (BIC), also known as Schwarz's Bayesian criterion of a `bbmFit` object from the value of the obtained log-likelihood stored in its `logLik` slot.

iter Extracts a subset of the iterations contained in a `bbmFit` object.

plot One plot for estimated abundances and one extra plot for each of the surveys with the fitting of total biomass and proportion of recruits.

Author(s)

Leire Ibaibarriaga & Sonia Sanchez

See Also

bbm, bbmFitresiduals, logLik, bbmFLPar, plot

Examples

```
# Load data
data(ane)

# Generate an object of bbmFit class (different alternatives)
new("bbmFit")               # empty object
slotNames(bbmFit())         # slots

# bbmFit: setting dimensions for stock.bio
bbmFit( stock.bio = FLQuant(dim=c(2,20,1,3,1,1), dimnames=list(age=1:2, year=1980:1999)))

# bbmFit: params class - FLPar with specific parameters for bbm function
bbmFit( params=bbmFLPar(years=dimnames(catch.ane)$year,
                        namesB=names(indicesB.ane), namesP=names(indicesP.ane)))
```

```
# Run assessment (output is of class bbmFit)
run <- bbm(catch.ane, indicesB=indicesB.ane, indicesP=indicesP.ane, control=control.ane, inits=inits.ane)
class(run)
run

# Plot
plot(run)


stock <- FLStock(catch.n=catch.ane, catch.wt=catch.ane*0+1)
units(stock@catch.wt) <- ''
stock@catch <- quantSums(stock@catch.n*stock@catch.wt)

newst <- stock + run # we must sum to the bbmFit object not to stock.bio(run)


# calculate residuals
residuals(run)

# log-Likelihood
logLik(run)

# AIC and BIC
AIC(run)

BIC(run)
```

`bbmFitresiduals` *bbmFitresiduals class*

Description

`bbmFitresiduals` class is used for storing the residuals for the fitted indices in the `bbm` function. This includes residuals for indices in total biomass (indicesB) and as proportions of recruits in biomass (indicesP).

Usage

```
bbmFitresiduals(object, ...)
```

Slots

residuals.B Residuals for indices in total biomass (indicesB), `FLQuants`.

residuals.P Residuals for indices as proportions of recruits in biomass (indicesP), `FLQuants`.

Accessors

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

Methods

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

plot Method to plot the Pearson residuals.

qqmath Method to do a qqplot of the Pearson residuals of survey indices.

Author(s)

Leire Ibaibarriaga & Sonia Sanchez

See Also

bbmFit, plot, qqmath

Examples

```
# Load data
data(ane)

# New element of class 'bbmFitresiduals' (empty)
bbmFitresiduals()

obj <- bbm( catch.ane, indicesB=indicesB.ane, indicesP=indicesP.ane, control=control.ane, inits=inits.ane)

res <- residuals(obj)
class(res)
slotNames(res)

# Starndardized residuals plots
plot(res)
qqmath(res)
```

---

bbmFLPar *Function to generate an FLPar for bbm function*

Description

This function generates an FLPar object with the input parametes required by the bbm function, given information on the years, the indices names and the number of iterations.

Usage

```
bbmFLPar(x = NULL, years, namesB, namesP, niter = 1, logscale = FALSE)
```

Arguments

x	Input numeric values for the parameters. If not set, then NA value is assigned.
years	Names of the years used for fitting.
namesB	Names of the indices of total biomass.
namesP	Names of the indices of proportion of recruits in biomass.
niter	Number of iterations.
logscale	Logical, if TRUE the parameters are in the scale used by the bbm model, otherwise, all parameters are in the linear scale. This is used for example, in the vcov matrix retuned from bbm function.

Value

An FLPar with the appropriate format for the bbm function.

Author(s)

Leire Ibaibarriaga & Sonia Sanchez.

See Also

FLPar, bbm

Examples

```
# Load data
data(ane)

years.ane <- dimnames(catch.ane)$year
niter.ane <- dim(catch.ane)[6]
namesB.ane <- names(indicesB.ane)
namesP.ane <- names(indicesP.ane)

# Generate population estimates, given some estimated parameters
pars <- bbmFLPar( years=years.ane, namesB=namesB.ane, namesP=namesP.ane, niter=niter.ane)
class(pars)
pars
```

---

calcIndices *Function to estimate fitted indices*

Description

This function estimates fitted indices, given information on growth, periods duration, catches and estimated parameters.

Usage

```
calcIndices(g, findicesB, findicesP, catch, inits)
```

Arguments

g	A vector with information on instantaneous rate of biomass decrease, g = M - G, for recruits (rec) and adults (adult).
findicesB	A vector with fraction of the year corresponding to each of the indicesB.
findicesP	A vector with fraction of the year corresponding to each of the indicesP.
catch	An FLQuant with catch information (for recruits and adults).
inits	An FLPar with parameter values for the parameters required by the bbm function.

Value

A list with information on estimated indices (FLQuants). The list has 2 elements: indicesB (for indices in biomass) and indicesP (for proportions of recruits in biomass).

Methods

Methods exist for various calculations based on the output class (FLPar). For details: ?FLPar.

Author(s)

Leire Ibaibarriaga & Sonia Sanchez.

See Also

FLPar, bbmFLPar, bbmControl

Examples

```
# Load data
data(ane)

args(calcIndices)

indices <- calcIndices( g=control.ane@g,
                findicesB=unlist(lapply( indicesB.ane, function(x) mean(range(x)[c('startf','endf')]))),
                findicesP=unlist(lapply( indicesP.ane, function(x) mean(range(x)[c('startf','endf')]))),
                        catch=catch.ane, inits=inits.ane )

class(indices)
slotNames(indices)
```

calcPop *Function to estimate abundances*

Description

This function estimates abundances in mass at age (for recruits and adults) by period, given information on growth, periods duration, catches and some additional values.

Usage

```
calcPop(g, f, catch, inits)
```

Arguments

g	A vector with information on instantaneous rate of biomass decrease, g = M - G, for recruits (rec) and adults (adult).
f	A vector with fraction of the year corresponding to each of the periods (determined by the period of the year of the different indices).
catch	An FLQuant with catch information (for recruits and adults).
inits	An FLPar with parameter values for the parameters required by the bbm function.

Value

A list with two elements: stock.bio, with information on estimated stock (FLQuant); and ok, an indicator on whether estimated parameters are valid (i.e. positive, ok==TRUE) or not (ok==FALSE).

Author(s)

Leire Ibaibarriaga & Sonia Sanchez.

See Also

FLPar, bbmFLPar

Examples

```
# Load required libraries
library(bbm)

# Load data
data(ane)

# Generate population estimates, given some estimated parameters
findicesB.ane <- unlist(lapply( indicesB.ane, function(x) mean(range(x)[c('startf','endf')])))
findicesP.ane <- unlist(lapply( indicesP.ane, function(x) mean(range(x)[c('startf','endf')])))
bioAge <- calcPop(g=control.ane@g,
                  f=periods( findicesB=findicesB.ane, findicesP=findicesP.ane)$f,
                  catch=catch.ane, inits=inits.ane)
```

```
class(bioAge)

# Check if valid ouput (i.e. positive biomass values)
bioAge$ok

# Estimates
bioAge$stock
```

---

createInits *Method createInits*

Description

Function for generating intial values for the parameters of the bbm function, given information on catches, survey indices and instantaneous rate of biomass decrease, g.

Usage

```
createInits(object, indicesB, indicesP, ...)

## S4 method for signature 'FLQuant,FLQuants,FLQuants'
createInits(object, indicesB, indicesP, findicesB, findicesP, g)

## S4 method for signature 'FLStock,ANY,ANY'
createInits(object, indicesB, indicesP, findicesB = NULL, findicesP = NULL, g)

## S4 method for signature 'FLQuant,FLIndices,FLIndices'
createInits(object, indicesB, indicesP, findicesB = NULL, findicesP = NULL, g)

## S4 method for signature 'FLQuant,FLQuant,FLQuant'
createInits(object, indicesB, indicesP, findicesB, findicesP, g)
```

Arguments

object	An FLQuant with catch information (for recruits and adults) or an FLStock.
indicesB	Abundance indices in total biomass (element of class: FLQuants, FLQuant or FLIndices) from surveys. Please assign a survey name to each index.
indicesP	Percentage of recruits in biomass (element of class: FLQuants, FLQuant or FLIndices) from surveys. Please assign a survey name to each proportion.
findicesB	A vector with fraction of the year corresponding to each of the indicesB.
findicesP	A vector with fraction of the year corresponding to each of the indicesP.
g	A vector with information on instantaneous rate of biomass decrease, g = M - G, for recruits (rec) and adults (adult).

Value

An object of class FLPar.

Methods

Methods exist for various calculations based on the output class (FLPar). For details: ?FLPar.

Author(s)

Leire Ibaibarriaga & Sonia Sanchez.

See Also

bbm, FLQuant, FLQuants, FLIndices, FLPar, bbmFLPar

Examples

```
# Load data
data(ane)


# Case: object='FLQuant'; indicesB=indicesP='FLQuants'
inits1 <- createInits( catch.ane,
                       indicesB=lapply( indicesB.ane, function(x) x@index),
                       indicesP=lapply( indicesP.ane, function(x) x@index),
                findicesB=unlist(lapply( indicesB.ane, function(x) mean(range(x)[c('startf','endf')]))),
                findicesP=unlist(lapply( indicesP.ane, function(x) mean(range(x)[c('startf','endf')]))),
                       g=control.ane@g )
class(inits1)

# Case: object='FLQuant'; indicesB=indicesP='ANY'

stock <- FLStock(catch.n=catch.ane, catch.wt=catch.ane*0+1)
units(stock@catch.wt) <- ''
stock@catch <- quantSums(stock@catch.n*stock@catch.wt)

inits2 <- createInits( stock, indicesB=indicesB.ane, indicesP=indicesP.ane,
                       g=control.ane@g )
class(inits2)

# Case: object='FLQuant'; indicesB=indicesP='FLIndices'

inits3 <- createInits( catch.ane,
                       indicesB=indicesB.ane, indicesP=indicesP.ane,
                       g=control.ane@g )
class(inits3)

# Case: object='FLQuant'; indicesB=indicesP='FLQuant'
inits4 <- createInits( catch.ane,
                     indicesB=indicesB.ane[[1]]@index, indicesP=indicesP.ane[[1]]@index,
                findicesB=c( depm=(indicesB.ane[[1]]@range[['startf']]+indicesB.ane[[1]]@range[['endf']])/2),
                findicesP=c( depm=(indicesP.ane[[1]]@range[['startf']]+indicesP.ane[[1]]@range[['endf']])/2),
```

```
                        g=control.ane@g )
class(inits4)

# Run assessment (with the different initial values)
run0 <- bbm(catch.ane, indicesB=indicesB.ane, indicesP=indicesP.ane, control=control.ane, inits=inits.ane)
run1 <- bbm(catch.ane, indicesB=indicesB.ane, indicesP=indicesP.ane, control=control.ane, inits=inits1)
run2 <- bbm(catch.ane, indicesB=indicesB.ane, indicesP=indicesP.ane, control=control.ane, inits=inits2)
run3 <- bbm(catch.ane, indicesB=indicesB.ane, indicesP=indicesP.ane, control=control.ane, inits=inits3)
namdel <- c("q_acoustic","psi_acoustic","xi_acoustic") # we will take only one of the indices --> need to delete the
control <- control.ane
control@param.fix <- control@param.fix[dimnames(control@param.fix)$params[!dimnames(control@param.fix)$params %
run4 <- bbm(catch.ane, indicesB=indicesB.ane[[1]]@index, indicesP=indicesP.ane[[1]]@index,
        findicesB=c( depm=(indicesB.ane[[1]]@range[['startf']]+indicesB.ane[[1]]@range[['endf']])/2),
        findicesP=c( depm=(indicesP.ane[[1]]@range[['startf']]+indicesP.ane[[1]]@range[['endf']])/2),
            control=control, inits=inits4)

# Plot assessed populations
biomass <- FLQuants()
runs <- paste("run",0:4,sep="")
names(runs) <- c('bc','run1','run2','run3','only_depm')
for (i in 1:length(runs)) biomass[[i]] <- quantSums(stock.bio(get(runs[i])))
names(biomass) <- names(runs)
plot( biomass)
```

`datasets` *bbm package datasets*

Description

Data of Bay of Biscay anchovy from Ibaibarriaga et al. (2008).

Example dataset for the classes defined in bbm package. At the moment there is only one dataset of Bay of Biscay anchvoy (ICES Subarea 8), with information on catches together with surveys observations and timing. These are the same data as used in Ibaibarriaga et al. (2008).

Dataset can be loaded by issuing the `data` command, like in `data(ane)`. and has defined the following objects:

- `catch.ane`, `FLQuant` Catch information in biomass for recruits (age 1 individuals) and adults.
- `indicesB.ane`, `FLIndices` Observations from surveys of total biomass, along with survey timing.
- `indicesP.ane`, `FLIndices` Observations from surveys of proportions of recruits in biomass, along with survey timing.
- `control.ane`, `bbmControl` Object with information on instantaneous rate of biomass decrease (g = M - G, `vector`) and information on whether bbm parameters are fixed or not (param.fix, `FLPar`).
- `inits.ane`, `FLPar` Initial values for the parameters of the bbm assessment model.

References

Ibaibarriaga et al. (2008). "A Two-Stage Biomass Dynamic Model for Bay of Biscay Anchovy: A Bayesian Approach." ICES J. of Mar. Sci. 65: 191-205.

See Also

bbm, FLQuant, FLQuants, FLStock, FLIndices, bbmControl, FLPar, bbmFLPar

Examples

```
data(ane)
ls()

is(catch.ane)
catch.ane

is(inits.ane)
inits.ane

is(control.ane)
control.ane

is(indicesB.ane)
indicesB.ane
is(indicesP.ane)
indicesP.ane
```

`periods` *Function to estimate the seasons*

Description

Given the fraction of the year when each of the indices are observed, this function estimates the number of seasons, the fraction of the year of each season and the season when each of the indices are observed.

Usage

```
periods(findicesB = NULL, findicesP = NULL)
```

Arguments

`findicesB`	A named `vector` with fraction of the year when each of the total biomass indices are observed.
`findicesP`	A named `vector` with fraction of the year when each of the proportion of recruits indices are observed.

Value

A `list` with four numeric elements: `nper`, with the number of seasons; `f` with the fraction of the year corresponding to each of the seasons; `perindicesB` with the season in which the index in total biomass is observed, this object is a named vector with one element per index; and `perindicesP` with the season in which the index of proportion of recruits is observed, this object is a named vector with one element per index.

Author(s)

Leire Ibaibarriaga & Sonia Sanchez.

Examples

```
# Load data
data(ane)

# Generate population estimates, given some estimated parameters
per <- periods( findicesB=unlist(lapply( indicesB.ane, function(x) mean(range(x)[c('startf','endf')]))),
           findicesP=unlist(lapply( indicesP.ane, function(x) mean(range(x)[c('startf','endf')]))))
class(per)

per$nper
per$f
per$perindicesB
per$perindicesB
```

`plot` *Plot of Pearson residuals of survey indices*

Description

Method to plot bbm fitting, if applied to a `bbmFit` object. The output are several plots. Firstly, the estimated abundance and next the fitted versus observed indices (with one plot for each survey). Note that in plots related to indices the yaxis doesn't has a scale. The visual is about the difference between the two lines, not about the value of each line, which in any case would be very difficult to assess visually.

Method to produce scatterplots of Pearson residuals of survey indices, if applied to a `bbmFitresiduals` object.

Usage

```
## S4 method for signature 'bbmFit,missing'
plot(x, y, ...)

## S4 method for signature 'bbmFitresiduals,missing'
plot(x, y = missing, auxline = "smooth", ...)
```

Arguments

x	An bbmFitresiduals object with the Pearson residuals.
y	Ignored.
...	Additional argument list.
auxline	A string defining the type of line to be added, by default uses 'smooth', a common alternative is to use 'r', a regression, or leave it empty ".

Value

If class(x)=='bbmFit', a plot with estimated abundances and one extra plot for each survey with fitted and observed indices.

If class(x)=='bbmFitresiduals', a plot with stardardized residuals.

Examples

```
# Load data
data(ane)

# Run the assessment
obj <- bbm( catch.ane, indicesB=indicesB.ane, indicesP=indicesP.ane, control=control.ane, inits=inits.ane)

# Plot the output
plot(obj)


# Residuals
res <- residuals(obj)
plot(res)
```

---

qqplot *Normal Q-Q Plot of surveys' indices*

Description

Method to produce qqplots of Pearson residuals against theoretical distribution of surveys' indices.

Usage

```
## S4 method for signature 'bbmFitresiduals,missing'
qqmath(x, data = missing, ...)
```

Arguments

x	An bbmFitresiduals object with the Pearson residuals.
data	Ignored.
...	Additional argument list that might never be used.

Value

A qqplot with stardardized log residuals against a theoretical distribution.

Examples

```
# Load data
data(ane)

# Run the assessment
obj <- bbm( catch.ane, indicesB=indicesB.ane, indicesP=indicesP.ane, control=control.ane, inits=inits.ane)

# Residuals
res <- residuals(obj)
qqmath(res)
```

---

simIndices *Method simIndices*

Description

Function to generate indices of total biomass and of proportion of recruits in biomass, given information on catches at age, instantaneous rate of biomass decrease (g = M - G) and values for the bbm parameters.

Usage

```
simIndices(object, ...)

## S4 method for signature 'FLQuant'
simIndices(object, g, inits, findicesB = NULL, findicesP = NULL)

## S4 method for signature 'FLStock'
simIndices(object, g, inits, findicesB = NULL, findicesP = NULL)
```

Arguments

`object`	An `FLQuant` with catch information (for recruits and adults).
`g`	A `vector` with information on instantaneous rate of biomass decrease, g = M - G, for recruits (rec) and adults (adult).
`inits`	An `FLPar` with initial values.
`findicesB`	A `vector` with period of the year of the different indices in total biomass. Optional parameter, if not set, then the survey is assumed to occur at the beggining of the year.
`findicesP`	A `vector` with period of the year of the different indices of proportion of recruits in mass. Optional parameter, if not set, then the survey is assumed to occur at the beggining of the year.

Value

A list with indices in biomass (Btot) and indices in proportion of recruits (Prec), both elements of the list are FLIndices.

Methods

Methods exist for various calculations based on the output class (`FLPar`). For details: `?FLPar`.

Author(s)

Leire Ibaibarriaga & Sonia Sanchez.

See Also

bbm, FLQuant, FLQuants, FLIndices, bbmControl, FLPar, bbmFLPar

Examples

```
# Load data
data(ane)

# Case:  object='FLQuant'
indices1 <- simIndices( catch.ane, g=control.ane@g, inits=inits.ane,
                findicesB=unlist(lapply( indicesB.ane, function(x) mean(range(x)[c('startf','endf')]))),
                findicesP=unlist(lapply( indicesP.ane, function(x) mean(range(x)[c('startf','endf')]))) )
class(indices1)
slotNames(indices1)


# Case:  object='FLStock'
stock <- FLStock(catch.n=catch.ane, catch.wt=catch.ane*0+1)
units(stock@catch.wt) <- ''
stock@catch <- quantSums(stock@catch.n*stock@catch.wt)

indices2 <- simIndices( stock, g=control.ane@g, inits=inits.ane,
                findicesB=unlist(lapply( indicesB.ane, function(x) mean(range(x)[c('startf','endf')]))),
                findicesP=unlist(lapply( indicesP.ane, function(x) mean(range(x)[c('startf','endf')]))) )
class(indices2)

# Run assessment with the alternative indices
run  <- bbm(catch.ane, indicesB=indicesB.ane, indicesP=indicesP.ane, control=control.ane, inits=inits.ane)
run1 <- bbm(catch.ane, indicesB=indices1$Btot, indicesP=indices1$Prec, control=control.ane, inits=inits.ane)
run2 <- bbm(catch.ane, indicesB=indices2$Btot, indicesP=indices2$Prec, control=control.ane, inits=inits.ane)

# Plot assessed populations
plot( FLQuants( bc=quantSums(run@stock.bio)[,,,1,], alt1=quantSums(run1@stock.bio)[,,,1,], alt2=quantSums(run2@s
```

Index