Package 'FLCore'

January 16, 2017

Title Core Package of FLR, Fisheries Modelling in R

Version 2.6.0.20161214

Date 2016-12-14

Description Core classes and methods for FLR, a framework for fisheries modelling and management strategy simulation in R. Developed by a team of fisheries scientists in various countries. More information can be found at http://flr-project.org/.

Depends R(>= 3.0),
MASS,
lattice

Imports methods,
utils,
Matrix,
grid,
stats,
graphics

License GPL (>= 2)

Repository flr

Collate 'genericMethods.R'
'uom.R'
'FLAccesors.R'
'classesArr.R'
'FLArray.R'
'FLQuant.R'
'FLQuantPoint.R'
'FLQuantDistr.R'
'FLPar.R'
'classesComp.R'
'classesLst.R'
'FLlst-class.R'
'FLComp.R'
'FLS.R'
'FLStock.R'
'FLStockLen.R'
'FLI.R'
'FLIndex.R'
'FLIndexBiomass.R'
'FLQuants.R'

'predictModel.R'
'FLBiol.R'
'FLModel.R'
'FLModelDeriv.R'
'FLModelSim.R'
'FLSR.R'
'operators.R'
'io.VPAsuite.R'
'io.FLStock.R'
'io.MFCL.R'
'io.ADMB.R'
'FLCohort.R'
'FLlst-methods.R'
'getPlural.R'
'io.FLIndices.R'
'FLGrowth.R'
'SRmodels.R'
'coerce.R'
'PlotDiagnostics.R'
'FLQuantJK.R'
'aliases.R'
'zzz.R'
'io.Adapt.R'
'io.VPA2Box.R'

LazyLoad Yes

LazyData No

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

Suggests testthat

RoxygenNote 5.0.1

R topics documented:

`FLCore-package` *Core package of FLR, fisheries modelling in R.*

Description

FLCore contains the core classes and methods for FLR, a framework for fisheries modelling and management strategy simulation in R. Developed by a team of fisheries scientists in various countries. More information can be found at http://flr-project.org/, including a development mailing list.

Details

Package:	FLCore
Version:	3.0
Date:	2009
Depends:	methods, R(>= 2.8.1), graphics, stats, stats4, grid, lattice
License:	GPL 2 or above
LazyLoad:	Yes
LazyData:	Yes
Built:	R 2.8.1; i686-pc-linux-gnu; 2009-01-22 12:08:46; unix

Classes:

FLArray	Class FLArray
FLBiol	Class FLBiol
FLBiols	Class FLBiols
FLCohort	Class FLCohort
FLCohorts	Class FLCohorts
FLComp	Class FLComp
FLIndex	Class FLIndex
FLIndices	Class FLIndices
FLModel	Class FLModel for statistical models
FLPar	Class FLPar
FLQuant	FLQuant class for numerical data
FLQuantPoint	Class FLQuantPoint
FLQuants	Class FLQuants
FLSR	Class FLSR
FLStoc	Class FLStock
FLStocks	Class FLStocks
FLlst	Class FLlst

Methods:

AIC	Akaike information criterion (AIC) method
Arith	Arithmetic methods for FLQuant objects
BIC	Bayesian information criterion (BIC) method
E	Method E
FLBiols-methods	Method FLBiols
	class
FLCohort-methods	Method FLCohort
FLCohorts-methods	Method FLCohorts
FLCore-package	FLCore package of the FLR system
FLIndex	Create FLIndex objects
FLIndices-methods	Method FLIndices
FLModel-methods	Method FLModel
FLPar-methods	Method FLPar
FLQuant-methods	Create FLQuant objects
FLQuantPoint-accesors	Method lowq
FLQuantPoint-methods	Method FLQuantPoint
FLQuants-methods	Method FLQuants
FLSR-methods	Method FLSR

```
                          Method median
mergeFLStock              Merging FLStock objects
model.frame,ANY-method
                          Method model.frame
name                      Accesor and replacement methods for complex S4
                          classes
names,ANY-method          Method names
nls,ANY,ANY,missing,missing,missing,missing,missing,missing,missing,missing,missing-meth
                          Method nls
ple4                      FLCore datasets
plot,ANY,ANY-method       Method plot
predict,ANY-method        Method predict
print,ANY-method          Method print
propagate                 Extend an FLQuant along the iter dimension
pv                        Population variability
qapply                    Method qapply
quant                     Method quant
quantSums                 Methods to compute sums, means and vars of
                          FLQuant objects
quantTotals               Method quantTotals
quantile,ANY-method       Method quantile
r                         Method for calculating intrinsic rate of
                          increase from an FLBiol object
range-methods             Method range
readVPAFile               Input/Output of FLR objects
revenue                   Method revenue
rgamma,ANY,ANY,ANY,ANY-method
                          Method rgamma
rlnorm,ANY,ANY,ANY-method
                          Method rlnorm
rnorm,ANY,ANY,ANY-method
                          Method rnorm
rpois,ANY,ANY,ANY-method
                          Method rpois
sd,FLModel,missing-method
                          Standard deviation of an FLModel object
setPlusGroup              Method setPlusGroup
show,ANY-method           Method show
sop                       Calculates the sum of products correction
splom,ANY,ANY-method      Method splom
spr0                      Method spr0
sr                        Stock-recruitment model function
ssb                       Method ssb
ssbpurec                  Method ssbpurec
ssn                       Method ssn
stripplot,ANY,ANY-method
                          Method stripplot
summary,ANY-method        Method summary
survprob                  Method for calculating survival probabilties
                          given mortaloty in the FLBiol object
sweep-methods             Sweep out FLQuant Summaries
transform,ANY-method      Transform elements of a complex FLR object
```

```
trim                        Trim FLR objects using named dimensions
units,ANY-method            units attribute for FLQuant objects
update,FLModel-method       Method update
upper                       Methods upper and lower
var,ANY,missing,missing,missing-method
                            Variance of an FLPar
window,ANY-method           Extract time (year) windows of an FLR object
```

Author(s)

FLR Team and various contributors. Initial design by Laurence T. Kell & Philippe Grosjean.

Maintainer: FLR Team <flr-team@flr-project.org>

References

Website at http://flr-project.org/

Kell L.T., Mosqueira I., Grosjean P., Fromentin J-M., Garcia D., Hillary R., Jardim E., Pastoors M., Poos J.J., Scott F. & Scott R.D. 2007. FLR: an open-source framework for the evaluation and development of management strategies. ICES J. of Mar. Sci. 20: 289-290.

AIC *Akaike information criterion (AIC) method*

Description

A method to calculate the Akaike information criterion (AIC) of an FLModel object from the value of the obtained log-likelihood stored in its `logLik` slot.

Generic function

AIC(object, k)

Method arguments

`object :` an object of class FLModel or of one that inherits from it.

`k :` numeric, the "penalty" per parameter to be used; the default k = 2 is the classical AIC.

Methods

signature(object=FLModel, k=numeric) : AIC of an FLModel object with an specified value for the "penalty".

signature(object=FLModel, k=missing) : AIC of an FLModel object with the default "penalty".

Author(s)

The FLR Team

See Also

AIC, logLik, FLModel

Examples

```
data(nsher)
AIC(nsher)
```

`aliases` *Short aliases for most FLCore classes construction methods*

Description

When working interactively, the naming convention of the FLCore classes can become tiresome. This set of short, lowercase aliases make calling the class construction methods a bit simpler.

Usage

```
flq(...)
flqp(...)
flqd(...)
flc(...)
flp(...)
fls(...)
flsl(...)
fli(...)
flib(...)
flsr(...)
flqs(...)
flcs(...)
flss(...)
flis(...)
flps(...)
flb(...)
flbs(...)
flms(...)
flmss(...)
pm(...)
```

Details

We recomend you use the full name of classes and methods when developing code that you intend to keep or distribute. A script written using these aliases can be later 'corrected' by substituting the aliases used in your editor.

The aliases' equivalences are as follows:

flq(...) For FLQuant(...)

flqp(...) For FLQuantPoint(...)

flqd(...) For FLQuantDistr(...)

flc(...) For FLCohort(...)

flp(...) For FLPar(...)

fls(...) For FLStock(...)

flsl(...) For FLStockLen(...)

fli(...) For FLIndex(...)

flib(...) For FLIndexBiomass(...)

flsr(...) For FLSR(...)

flqs(...) For FLQuants(...)

flcs(...) For FLCohorts(...)

flss(...) For FLStocks(...)

flis(...) For FLIndices(...)

flps(...) For FLPars(...)

flb(...) For FLBiol(...)

flbs(...) For FLBiols(...)

flms(...) For FLModelSim(...)

flmss(...) For FLModelSims(...)

pm(...) For predictModel(...)

Value

An object of the requested class

Author(s)

The FLR Team

See Also

FLQuant, FLQuantPoint, FLQuantDistr, FLCohort, FLPar, FLStock, FLStockLen, FLIndex, FLIndexBiomass, FLSR, FLQuants, FLCohorts, FLStocks, FLIndices, FLPars, FLBiol, FLBiols, FLModelSim, FLModelSims, predictModel

Examples

```
flq <- flc(1:10, units="kg")
```

---

apply *Method apply*

Description

Functions can be applied to margins of an FLQuant array using this method. In contrast with the standard method, dimensions are not collapsed in the output object.

FUN in the case of an FLQuant must collapse at least one dimension when applied over an array.

For further details see apply.

Generic function

apply(X,MARGIN,FUN)

Method arguments

`X` **:** an object of a class for which this method has been defined.

`MARGIN` **:** a vector giving the subscripts which the function will be applied over. 1 indicates rows, 2 indicates columns, c(1,2) indicates rows and columns, etc.

`FUN` **:** the function to be applied.

`...` **:** optional arguments to FUN.

Methods

signature(X=ANY,MARGIN=missing,FUN=missing) : New S4 generic based on base::apply

signature(X=FLQuant,MARGIN=numeric,FUN=function) : apply a given function over the selected dimensions of an `FLQuant`. Returns an object of class `FLQuant`.

Author(s)

The FLR Team

See Also

apply

Examples

```
flq <- FLQuant(rlnorm(100), dim=c(10,20,1,1,1,5))
apply(flq, 1, sum)
apply(flq, 2:6, sum)
```

`Arith` *Arithmetic methods for FLQuant objects*

Description

The `Arith` group of methods, comprising addition, substraction, product, division, exponentiation, and integer division (`+`, `-`, `*`, `/`, `^`, `%%` and `%/%`). These methods work exactly as in an object of class `array`, but always return an `FLQuant` object.

Generic function

Arith(e1,e2)

Methods

signature(e1=FLQuant,e2=FLQuant) : Operations between two FLQuant objects

Author(s)

The FLR Team

See Also

`Arithmetic`, `Arith`

Examples

```
flq <- FLQuant(rnorm(10), dim=c(2,5))
fl2 <- FLQuant(2, dim=c(2,5))
flq*fl2
flq/fl2
```

`as.data.frame` *Method as.data.frame*

Description

This method converts an FLQuant or any other FLR object composed of FLQuants into a `data.frame`.

For a single FLQuant, the `data.frame` returned has 7 columns: `quant`, `year`, `unit`, `season`, `area`, `iter` and `data`. The last column contains the actual values stored in the original object, while the first six contain the corresponding dimensions. The `year` and `data` columns are of class numeric, while the other five are of class factor.

When converting an `FLCohort` object, the `year` column is substituted by `cohort`.

The `data.frame` returned for complex objects, i.e. those the inherit from class `FLComp`, has an extra column, `slot`, that holds the name of the slot in the original object.

The data.frame obtained from an `FLQuants` object also has an extra column, named `qname`, that refers to the name of each `FLQuant` object in the list. This column is named `cname` when an `FLCohorts` object is converted.

Objects of class FLQuants can also be converted into a wide-format table, where data from the list elements are placed in separate colums, using model.frame,FLlst-method.

Generic function

as.data.frame(x, row.names, optional)

Methods

signature(x=FLQuant,row.names=ANY,optional=ANY) : Converts an FLQuant into a data.frame

signature(x=FLComp,row.names=missing,optional=missing) : Converts objects of any class inheriting from FLComp into a data.frame

signature(x=FLQuants,row.names=missing,optional=missing) : Converts objects of class FLQuants into a data.frame

signature(x=FLCohorts,row.names=missing,optional=missing) : Converts objects of class FLCohorts into a data.frame

signature(x=FLPar,row.names=ANY,optional=ANY) : Converts objects of class FLPar into a data.frame

signature(x=FLCohort,row.names=ANY,optional=ANY) : Converts objects of class FLCohort into a data.frame

Author(s)

The FLR Team

See Also

as.data.frame, model.frame, model.frame,FLlst-method

Examples

```
data(ple4)
fdf <- as.data.frame(catch.n(ple4))
head(fdf)
summary(fdf)

sdf <- as.data.frame(ple4)
head(sdf)
```

---

asOld *Old S3 coercion methods*

Description

These methods convert or coerce an object of a given class into another class. They follow the S3 syntax for coercion methods and are being slowly substituted by `as()` (see `coerce`.

Generic function

as.FLIndex(object) Convert to an `FLIndex`

as.FLSR(object) Convert to an `FLSR`

as.FLQuant(x) Convert to an `FLQuant`

Methods

signature(object=FLStock) : Describe method

signature(x=array) : Describe method

signature(x=matrix) : Describe method

signature(x=FLQuant) : Describe method

signature(x=vector) : Describe method

signature(x=data.frame) : Describe method

Author(s)

The FLR Team

See Also

FLComp

Examples

```
data(ple4)
is(ple4)
```

BIC *Bayesian information criterion (BIC) method*

Description

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

Generic function

BIC(object)

Methods

signature(object=FLModel) : BIC of an FLModel object with an specified value for the "penalty".

signature(object=FLModel, k=missing) : AIC of an FLModel object for which no value of "penalty" is specified, thus k=2.

Author(s)

The FLR Team

See Also

BIC, AIC, FLModel, logLik

Examples

```
data(nsher)
BIC(nsher)
```

bkey *Generate key for bubbles plot*

Description

`bkey` generates the bubble scale that shows on bubbles plot for `FLQuants` objects. It is not intended to be used directly by users but you may try.

Generic function

bkey(object)

Methods

signature(object=list): The list must have at least the elements `data`, `cex`, `bub.col` and `bub.scale`.

Author(s)

The FLR Team

See Also

FLComp

`bubbles` *Bubbles plot*

Description

This method plots three dimensional data like matrices by age and year or age-class, very common in fisheries. The bubbles are proportional to the values on the matrix. Note that `bubbles` accept an argument `bub.scale` to control the relative size of the bubbles. Positive and negative values have separate colours.

Generic function

bubbles(x, data)

Methods

signature(x=formula, data=FLQuant) : Produce bubbles plot of the object.

signature(x=formula, data=FLQuants) : Produce a lattice of bubbles plot of the objects. Commonly used to plot residuals of VPA assessments.

signature(x=formula, data=FLCohort) : Produce bubbles plot of the object.

signature(x=formula, data=data.frame) : Produce bubbles plot of the object.

Author(s)

The FLR Team

See Also

lattice, `FLQuant`, `FLQuants`,`FLCohort`

Examples

```
data(ple4)
bubbles(age~year, data=catch.n(ple4))
bubbles(age~year, data=catch.n(ple4), bub.scale=5)
bubbles(age~cohort, data=FLCohort(catch.n(ple4)), bub.scale=5)

qt01 <- log(catch.n(ple4)+1)
qt02 <- qt01+rnorm(length(qt01))
flqs <- FLQuants(qt01=qt01, qt02=qt02)
bubbles(age~year|qname, data=flqs, bub.scale=1)

qt03 <- FLQuant(rnorm(100),dimnames=list(age=as.character(1:10),year=as.character(1:10)))
bubbles(age~year, data=qt03, bub.scale=7, col=c("black","red"))
```

ccplot *Catch-curves plot*

Description

Catch-curves are essential to explore the mortality carried out on a stock. It shows the trends on different cohorts by age.

Generic function

ccplot(x,data)

Methods

signature(x=formula,data=FLCohort) : Trends on cohorts by age.

Author(s)

The FLR Team

See Also

FLComp

Examples

```
data(ple4)
ccplot(data~age, data=FLCohort(ple4@catch.n), type="l")
```

coerce *Method coerce*

Description

Coercion methods for various sets of classes are generated using the `coerce` function. Users should call the corresponding generated `as()` method, with arguments equal to the object to coerce and the name of the class to convert to.

Coercion combinations work by transferring or transforming relevant slots from the original object and placing them in a new object of the target class. The descriptions below document how slots for each pair of classes are transfered or transformed. In all cases the `name` and `desc` slots are simply copied accross.

Generic function

coerce(from, to, strict)

Methods

from=FLBiol, to=FLStock: n to stock.n, wt to stock.wt, m to m,fec to mat,spwn to m.spwn

from=FLStock, to=FLBiol: stock.n to n, stock.wt to wt, m to m,mat to fec,m.spwn to spwn catch.wt, catch to catch, landings.n to landings.n, landings.wt to landings.wt, landings to landings, discards.n to discards.n, discards.wt to discards.wt, discards to discards, name to name

from=FLQuant,to=FLCohort An FLCohort object is created from the year-structured data in an FLQuant. See FLCohort for a description of the exact procedure.

from=FLCohort,to=FLQuant The previous calculation is reversed and an FLQuant is returned.

from=FLlst,to=list An standard R list object is created from an FLlst by dropping the extra attributes.

Author(s)

The FLR Team

See Also

FLComp

Examples

```
data(ple4)
flb <- as(ple4, 'FLBiol')
```

---

computeCatch *Methods to compute total catch, landings, discards and stock biomass*

Description

These methods compute the total catch, landings, discards and stock biomass from the quant-structured values in numbers and weight per individual. The calculation for discards, landings and stock involves the product of the landings/discards/stock in numbers (landings.n, discards.n or stock.n) by the individual weight-at-quant (landings.wt, discards.wt or stock.wt), as in

$$L = L_n * L_{wt}$$

By selecting slot="catch", computeCatch can calculate in the same way the total catch from the catch-at-quant and weight in the catch. Those two values (in slots catch.n and catch.wt can also be calculated by specifying slot="n" and slot="wt" respectively. Calling computeCatch with option slot="all" will carry out the three calculations. In this case, the returned object will be of class FLQuants, with elements names catch, catch.n and catch.wt, which can then be passed directly to the catch<- replacement method.

Generic function

computeCatch(object, ...)

computeLandings(object, ...)

computeDiscards(object, ...)

computeStock(object, ...)

Method arguments

`object :` an object of a class for which this method has been defined.

`slot :` a character vector to select the calculation to perform in `computeCatch`. One of "n", "wt", "all" or "catch", the default value. The later will compute the total catch (for slot `catch`) from `catch.n` and `catch.wt`

`na.rm :` a logical indicating whether `NA`s should be deleted from the sums. Defaults to `TRUE`.

Methods

signature(object=FLStock) : computation on an `FLStock` object.

signature(object=FLIndex) : computation on an `FLIndex` object.

Author(s)

The FLR Team

See Also

FLComp

Examples

```
data(ple4)
summary(computeLandings(ple4))
landings(ple4) <- computeLandings(ple4)
catch(ple4) <- computeCatch(ple4, slot="all")
```

---

createFLAccesors *Create accesor methods for a given class*

Description

This function creates a complete set of standard S4 class accessors and replacers. Not intended for direct use.

Usage

```
createFLAccesors(class, exclude = character(1), include=missing)
```

Arguments

`class`	name of the class
`exclude`	Slot names to exclude
`include`	Slot names to include

Author(s)

The FLR Team

cv *Coefficient of Variation of FLR objects with multiple iterations*

Description

The Coefficient of Variation of an object with mutiple iterations along the sixth (`iter`) dimension can be calculated using `cv()`. An object of the same class, with length=1 on the sixth dimension, will be returned.

CV of x is calculated as $\frac{sd(x)}{\hat{x}}$.

For objects of class `FLModel`, cv returns the result of

$$\frac{\sqrt{diag(\Sigma)}}{\Theta}$$

where Σ is the variance-covariance matrix of the Θ parameter set.

Generic function

cv(object)

Methods

signature(object=FLQuant) : Works along the `iter` dimension of an `FLQuant`

Author(s)

The FLR Team

See Also

FLComp

Examples

```
flq <- FLQuant(rnorm(200, 5, 10), dim=c(5,10), iter=100)
cv(flq)
```

data *FLCore datasets*

Description

- ple4A dataset of North Sea (ICES Area IV) plaice catch, yield, landings, discards, natural mortality, weight-at-age and maturity, together with the VPA estimated abundances and fishing mortalities, contained in an FLStock object.
- ple4sexA dataset of North Sea (ICES Area IV) plaice disaggregated by sex catch, yield, landings, discards, natural mortality, weight-at-age and maturity, together with the VPA estimated abundances and fishing mortalities, contained in an FLStock object.

- ple4.indexA dataset of North Sea (ICES Area IV) plaice survey catch per unit effort, index and index variance, contained in an FLIndex object.
- ple4.indicesA dataset of two North Sea (ICES Area IV) plaice survey catch per unit effort, index and index variance, contained in an FLIndices object.
- ple4.indexA dataset of North Sea (ICES Area IV) plaice survey catch per unit effort, index and index variance, contained in an FLIndex object.
- ple4.indicesA dataset of two North Sea (ICES Area IV) plaice survey catch per unit effort, index and index variance, contained in an FLIndices object.
- ple.biolA dataset of North Sea plaice population, numbers, natural mortality, mass and fecundity-at-age, contained in an FLBiol object.
- nsherA dataset of class `FLStock` for autumn spawning North Sea herring.
- nsher.srA dataset of North Sea autumn spawning herring stock and recruitment relationship. Fitted using the 'Ricker' model.
- nsher.biolA dataset of class `FLBiol` for North Sea herring. Datasets can be loaded by issuing the `data` command, like in `data(ple4)`.

References

ICES.

See Also

FLStock, FLSR, FLIndex, FLStock, FLIndex, FLBiol

`dimnames<-` *Modify dimnames of an FLQuant*

Description

The `dimnames<-` method for objects of class `FLQuant` modifies the `dimnames` attribute. In contrast with the method for class `array`, an incomplete named list of dimension names can be provided. Only the relevant dimensions will be modify.

It is posible to modify the name of the first dimension (by default `quant`) using this method.

Generic function

dimnames<-(x,value)

Methods

signature(x=FLQuant, value=list) : Modify FLQuant dimnames.

signature(x=FLStock, value=list) : Modify dimnames of all FLQuant slots inside an FLStock object.

Author(s)

The FLR Team

See Also

dimnames, FLQuant, array

Examples

```
flq <- FLQuant(rnorm(80), dim=c(4,10,2))
dimnames(flq) <- list(unit=c('male', 'female'))
# This modifies both dimnames and dimnames name
dimnames(flq) <- list(age=0:3)
dimnames(flq)
```

---

dims *List with information on object dimensions*

Description

dims return a named list with information on the dimensions and dimension names of a given object obj. The list returned could be extended in the future and currently contains, depending on the class of the object, some of the following:

quant Length of the first dimensions, i.e. number of ages, lengths, etc.

min First quant

max Last quant

year Number of years

minyear First year in series

maxyear Last year in series

cohort Number of cohorts

mincohort First cohort in series

maxcohort Last cohort in series

unit Lengt of the third (unit) dimension

season Lengt of the fourth (season) dimension

area Lengt of the fifth (area) dimension

iter Lengt of the sixth (iter) dimension

Values in the returned list are of class numeric, unless dimnames are strings with no numeric translation, in which case the result is NA.

Please note that the name of the first element in the returned list changes with the name of the first dimension on the input object. Use quant to obtain the name and extract the relevant element from the result list.

Generic function

dims(obj)

Methods

signature(obj=FLQuant) : Describe method

signature(obj=FLComp) : Describe method

Author(s)

The FLR Team

See Also

dimnames, FLQuant

Examples

```
flq <- FLQuant(rnorm(96), dim=c(3,8,1,4), quant='age')
dims(flq)

# Number of seasons
dims(flq)$season

# Length of first dimension
dims(flq)[[quant(flq)]]
```

`evalPredictModel` *Evaluates a predictModel slot inside the object cointaining it*

Description

Models in objects of the predictModel class can make use of slots and methods of the FLR class in which it is contained as a slot. This function can be used by methods wishing to evaluate a single 'predictModel' slot in the context of the class it isd part of.

Usage

```
evalPredictModel(object, slot)
```

Arguments

`object`	The FLR S4 object holding the 'predictModel' slot.
`slot`	The name of the slot to be evaluated, as a character.

Value

The result of evaluating the model, usually an 'FLQuant'

Author(s)

The FLR Team

See Also

predictModel

expand *Trim FLR objects using named dimensions*

Description

Need to add

Generic function

expand(x)

Methods

signature(x=FLArray) : Describe method

signature(x=FLComp) : Describe method

signature(x=FLStock) : Describe method

Author(s)

The FLR Team

See Also

FLComp

Examples

```
data(ple4)

expand(ple4, year=1957:2013)
```

Extract *Extract or Replace Parts of an FLR Object*

Description

Operators acting on FLQuant, FLCohort, FLPar, FLComp, and derived classes to extract or replace sections of an object.

Please note the difference between referencing sections of an object by position, using values of class `numeric`, or by dimnames, of class `character`. See examples below.

All classes that are derived from `FLComp` (for example, `FLStock` and `FLBiol`) can be subset along the six dimensions of their `FLQuant` slots.

Classes that are derived from `FLlst` (for example, `FLStocks` and `FLBiols`) can be subset in a similar way to ordinary list objects.

Generic function

[x,i,j,drop]

[<-(x,i,j,value)

[[<-(x,i,j,value)

\$<-(x,name,value)

Methods

signature(x=FLQuant,i=ANY,j=ANY,drop=missing) : Returns an `FLQuant` object, subset along any of the six dimensions (quant, year, unit, season, area and iter).

signature(x=FLPar,i=ANY,j=ANY,drop=missing) : Subset an `FLPar` object.

signature(x=FLStock,i=ANY,j=ANY,drop=missing) : Returns an `FLStock` where all the `FLQuant` slots have been subset by quant, year, unit, season, area and iter.

signature(x=FLBiol,i=ANY,j=ANY,drop=missing) : Returns an `FLBiol` where all the `FLQuant` slots have been subset by quant, year, unit, season, area and iter.

signature(x=FLCohort,i=ANY,j=ANY,drop=missing) : Describe method

signature(x=FLlst,i=ANY,j=missing,value=missing) : Returns the specified element of the list. For example, for an `FLStocks` object, stocks[[1]] will return a single `FLStock`.

signature(x=FLlst,name=character,value=missing) : Describe method

Author(s)

The FLR Team

See Also

Extract

Examples

```
flq <- FLQuant(rnorm(50), dimnames=list(age=1:5, year=1990:2000, season=1:4))
flq[1,]
flq[,1:5]
flq[,'1990']
flq[1:2,,,c(1,3)]
```

---

fbar *Calculates mean harvest rate or fishing mortality*

Description

The mean harvest rate of fishing mortality

Usage

```
fbar(object, ...)
```

Arguments

object	An FLStock or FLBiol object
...	Any extra arguments, currently unused

Details

The average fishing mortality for the years between *minfbar* and *maxfbar*, as found in the range slot, is returned.

Value

An object of class FLQuant

Author(s)

FLR Team

Examples

```
data(ple4)
fbar(ple4)
```

FLArray *Class FLArray*

Description

A basic 6D array class. No objects of this class are created in FLCore, as it is used only for method inheritance.

Slots

.Data Internal S4 data representation, of class array.

Validity

Dimensions: Array must have 6 dimensions

Content: Array must be of class numeric

Author(s)

The FLR Team

See Also

FLQuant, FLCohort

FLBiol *Class FLBiol*

Description

A class for modelling age / length or biomass structured populations.

Details

The FLBiol class is a representation of a biological fish population. This includes information on abundances, natural mortality and fecundity.

Slots

n Numbers in the population. FLQuant.

m Mortality rate of the population. FLQuant.

wt Mean weight of an individual. FLQuant.

mat predictModel.

fec predictModel.

rec predictModel.

spwn Proportion of time step at which spawning ocurrs. FLQuant.

name Name of the object. character.

desc Brief description of the object. character.

range Named numeric vector describing the range of the object. numeric.

Accesors

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 stored in any slot.

Validity

Dimensions All FLQuant slots must have iters equal to 1 or 'n'.

Iters The dimname for iter[1] should be '1'.

Dimnames The name of the quant dimension must be the same for all FLQuant slots.

Author(s)

The FLR Team

See Also

as.FLBiol, as.FLSR, coerce, plot, ssb catch.n,FLBiol-method

Examples

```
# An FLBiol example dataset
data(ple4.biol)

summary(ple4.biol)
```

`FLBiols` *Class FLBiols*

Description

A list of `FLBiol` objects.

Slots

.Data Internal S4 data representation, of class `list`.

desc As textual description of the object contents

lock Can the object be extended/trimmed? `TRUE` or `FALSE`.

names A character vector for the element names

Extends

FLlst list vector

Constructor

The `FLBiols(object, ...)` constructor method allows simple creation of new FLBiols objects with the methods described below.

signature(object=ANY) : Returns an FLBiols object

signature(object=missing) : Returns an empty FLBiols object

signature(object=list) : Returns an FLBiols object

Methods

All methods are inherited.

Author(s)

The FLR Team

See Also

FLlst, list, vector

flc2flq *Coerce FLCohort into FLQuant.*

Description

Coerces FLCohort objects into FLQuant objects. It's also implemented with seAs, to be used like as(flcobject, "FLQuant") and it will be deprecated in the near future.

Generic function

flc2flq(object)

Methods

signature(object=FLCohort) : Coerce FLCohort into FLQuant.

Author(s)

The FLR Team

See Also

FLComp

Examples

```
data(ple4)
flc <- FLCohort(catch.n(ple4))
flq <- flc2flq(flc)
all.equal(flq, catch.n(ple4))
```

FLCohort *Class FLCohort*

Description

This class represents cohorts in columns. It simply shifts the typical matrix representation where cohorts are found on the diagonals, into a matrix where cohorts are found in columns. It is very usefull for all analysis that want to make use of cohorts instead of years.

Slots

.Data Internal S4 data representation. array.

units The data units in some understandable metric. character

Extends

FLArray array

Constructor

The `FLCohort(object)` constructor method allows simple creation of new FLCatch with the methods described below.

signature(object=FLQuant) : Creates a FLCohort object from a FLQuant object. It simply shifts the matrix so that cohorts instead of years are located in the columns (second dimensions) of the array.

signature(object=array) : Creates a FLCohort object from an array.

signature(object=missing) : Creates a empty FLCohort object.

Methods

[(base) : Subset method

`signature(x=FLCohort,i=ANY,j=ANY,drop=missing)`

as.data.frame(base) : Coerce to data.frame.

`signature(x=FLCohort,row.names=ANY,optional=ANY)`

bubbles(FLCore) : Bubbles plot.

`signature(x=formula,data=FLCohort)`

ccplot(FLCore) : Catch curves plot.

`signature(x=formula,data=FLCohort)`

flc2flq(FLCore) : Coerce to FLQuant (deprecated).

`signature(object=missing)`

FLCohort(FLCore) : Creator method based on FLQuant objects.

`signature(object=FLQuant)`

plot(graphics) : Simple plot

`signature(x=FLCohort,y=ANY)`

quant(FLCore) : Extract the quant dimension definition.

`signature(object=missing)`

trim(FLCore) : Subset based on limiting dimnames.

`signature(object=missing)`

units(base) : Extract the information about data units.

`signature(x=missing)`

units<-(base) : Replace data units information.

`signature(x=FLCohort,value=character)`

xyplot(lattice) : Lattice's xyplot method.

`signature(x=formula,data=FLCohort)`

Author(s)

The FLR Team

See Also

[, as.data.frame, bubbles, ccplot, FLCohort,FLQuant-method, flc2flq, plot, quant, trim, units, units<-,FLCohort,character-method, xyplot, array

Examples

```
data(ple4)
flq <- catch.n(ple4)
flc <- FLCohort(flq)
plot(trim(flc, cohort=1960:2000))
```

FLCohorts *Class FLCohorts*

Description

FLCohorts is a class that extends list through FLlst but implements a set of features that give a little bit more structure to list objects. The elements of FLCohorts must all be of class FLCohort. It implements a lock mechanism that, when turned on, does not allow the user to increase or decrease the object length.

Slots

.Data The data. list

names Names of the list elements. character

desc Description of the object. character

lock Lock mechanism, if turned on the length of the list can not be modified by adding or removing elements. logical

Extends

FLlst list vector

Constructor

The FLCohorts(object, ...) constructor method allows simple creation of new FLCatch with the methods described below.

signature(object=ANY) : Returns an FLCohorts object

signature(object=missing) : Returns an empty FLCohorts object

signature(object=list) : Returns an FLCohorts object

Methods

***(base) :** Describe method

signature(e1=FLCohorts,e2=FLCohorts)

Arith(methods) : Describe method

signature(e1=FLCohorts,e2=FLCohorts)

as.data.frame(base) : Describe method

signature(x=FLCohorts,row.names=missing,optional=missing)

bubbles(FLCore) : Describe method

signature(x=formula,data=FLCohorts)

catch<-(FLCore) : Describe method

`signature(object=FLStock,value=FLCohorts)`

iter(FLCore) : Describe method

`signature(object=missing)`

model.frame(stats) : Describe method

`signature(formula=missing)`

show(methods) : Describe method

`signature(object=missing)`

summary(base) : Describe method

`signature(object=missing)`

xyplot(lattice) : Describe method

`signature(x=formula,data=FLCohorts)`

Author(s)

The FLR Team

See Also

*, Arith, as.data.frame, bubbles, catch<-, iter, model.frame, show, summary, xyplot, FLlst, list

---

`FLComp` *Class FLComp*

Description

A virtual class that forms the basis for most FLR classes composed of slots of class `FLQuant`. No objects of this class can be constructed.

Validity

Dimensions All FLQuant slots must have iters equal to 1 or 'n'.

Iters The dimname for iter[1] should be '1'.

Dimnames The name of the quant dimension must be the same for all FLQuant slots.

Slots

name A character vector for the object name.

desc A textual description of the object contents.

range A named numeric vector with various values of quant and year ranges, plusgroup, fishing mortality ranges, etc.

Author(s)

The FLR Team

See Also

[, [<-, as.data.frame, iter, propagate, qapply, summary, transform, trim, units,FLComp-method, units<-,FLComp,list-method, window

FLCore-internal *FLCore-internal*

Description

FLCore-internal

Author(s)

The FLR Team

FLI *Class FLI*

Description

A VIRTUAL class that holds data and parameters related to abundance indices.

Slots

type Type of index (`character`).

distribution Statistical distribution of the index values (`character`).

index Index values (`FLQuant`).

index.var Variance of the index (`FLQuant`).

catch.n Catch numbers used to create the index (`FLQuant`).

catch.wt Catch weight of the index (`FLQuant`).

effort Effort used to create the index (`FLQuant`).

sel.pattern Selection pattern for the index (`FLQuant`).

index.q Catchability of the index (`FLQuant`).

name Name of the stock (`character`).

desc General description of the object (`character`).

range Range of the object (`numeric`)

Accesors

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 stored in any slot.

Validity

Dimensions All FLQuant slots must have iters equal to 1 or 'n'.

Iters The dimname for iter[1] should be '1'.

Dimnames The name of the quant dimension must be the same for all FLQuant slots.

Author(s)

The FLR Team

See Also

computeCatch, dims, iter, plot, propagate, summary, transform, trim, window, FLComp

---

`FLIndex` *Class FLIndex*

Description

A class that holds data and parameters related to abundance indices.

Slots

type Type of index (`character`).

distribution Statistical distribution of the index values (`character`).

index Index values (`FLQuant`).

index.var Variance of the index (`FLQuant`).

catch.n Catch numbers used to create the index (`FLQuant`).

catch.wt Catch weight of the index (`FLQuant`).

effort Effort used to create the index (`FLQuant`).

sel.pattern Selection pattern for the index (`FLQuant`).

index.q Catchability of the index (`FLQuant`).

name Name of the stock (`character`).

desc General description of the object (`character`).

range Range of the object (`numeric`)

Accesors

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 stored in any slot.

Validity

Dimensions All FLQuant slots must have iters equal to 1 or 'n'.

Iters The dimname for iter[1] should be '1'.

Dimnames The name of the quant dimension must be the same for all FLQuant slots.

Author(s)

The FLR Team

See Also

computeCatch, dims, iter, plot, propagate, summary, transform, trim, window, FLComp

Examples

```
fli <- FLIndex(index=FLQuant(rnorm(8), dim=c(1,8)), name="myTestFLindex")
summary(fli)
index(fli)
```

FLIndexBiomass *Class FLIndexBiomass*

Description

A class that holds data and parameters related to biomass abundance indices.

Slots

distribution Statistical distribution of the index values (`character`).

index Index values (`FLQuant`).

index.var Variance of the index (`FLQuant`).

catch.n Catch numbers used to create the index (`FLQuant`).

catch.wt Catch weight of the index (`FLQuant`).

effort Effort used to create the index (`FLQuant`).

sel.pattern Selection pattern for the index (`FLQuant`).

index.q Catchability of the index (`FLQuant`).

name Name of the stock (`character`).

desc General description of the object (`character`).

range Range of the object (`numeric`)

Accesors

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 stored in any slot.

Validity

Dimensions All FLQuant slots must have iters equal to 1 or 'n'.

Iters The dimname for iter[1] should be '1'.

Dimnames The name of the quant dimension must be the same for all FLQuant slots.

Author(s)

The FLR Team

See Also

computeCatch, dims, iter, plot, propagate, summary, transform, trim, window, FLComp

Examples

```
idx <- FLIndexBiomass(index=FLQuant(1:10, quant='age'))

data(ple4)
ida <- FLIndexBiomass(index=ssb(ple4),
  catch.n=catch.n(ple4))
```

`FLIndices` *Class FLIndices*

Description

`FLIndices` is a class that extends `list` through `FLlst` but implements a set of features that give a little bit more structure to list objects. The elements of `FLIndices` must all be of class `FLIndex`. It implements a lock mechanism that, when turned on, does not allow the user to increase or decrease the object length.

Slots

.Data The data. `list`.

names Names of the list elements. `character`.

desc Description of the object. `character`.

lock Lock mechanism, if turned on the length of the list can not be modified by adding or removing elements. `logical`.

Extends

FLlst list

Constructor

The `FLIndices(object, ...)` constructor method allows simple creation of new FLIndices with the methods described below.

signature(object=ANY) : Returns an FLIndices object

signature(object=missing) : Returns an empty FLIndices object

signature(object=list) : Returns an FLIndices object

Methods

All methods are inherited.

Author(s)

The FLR Team

See Also

FLlst, list

Examples

```
data(ple4.index)
flis <- FLIndices(INDa=ple4.index, INDb=window(ple4.index, end=2000))
```

`FLlst` *Class FLlst*

Description

`FLlst` is a class that extends `list` but implements a set of features that give a little bit more structure to list objects. First the elements of `FLlst` must all be of the same class. Second it implements a lock mechanism that, when turned on, does not allow the user to increase or decrease the object length.

Slots

.Data The data. `list`.

names Names of the list elements. `character`.

desc Description of the object. `character`.

lock Lock mechanism, if turned on the length of the list can not be modified by adding or removing elements. `logical`.

Extends

list

Constructor

The `FLlst(object)` constructor method allows simple creation of new FLlst with the methods described below.

signature(object=ANY) : Returns an `FLlst` object with the provided elements if they are all o the same class.

signature(object=missing) : Returns an empty `FLlst` object

signature(object=list) : Returns and `FLlst` object with the input list at its core

Methods

[(base) : Select method.

```
signature(x=FLlst,i=ANY,j=missing,drop=missing)
```

[<-(base) : Replacement method for elements.

```
signature(x=FLlst,i=ANY,j=missing,value=ANY)
```

[[<-(base) : Replacement method within elements.

```
signature(x=FLlst,i=ANY,j=missing,value=missing)
```

$<-(base) : Replacement method for elements.

```
signature(x=FLlst,name=character,value=missing)
```

coerce(methods) : Coerce method.

```
signature(from=FLlst,to=list,strict=missing)
```

lapply(base) : `lapply` implemented for `FLlst` objects.

```
signature(X=missing,FUN=missing)
```

window(stats) : Selects a set of years from all elements at once.

```
signature(x=missing)
```

Author(s)

The FLR Team

See Also

[, [<-, [[<-, $<-, coerce, lapply, window, list

Examples

```
fll01 <- new("FLlst", list(a=1:10, b=10:20))
fll02 <- new("FLlst", list(1:10, 10:20), names=c("a","b"))
fll03 <- FLlst(a=1:10, b=10:20)
fll04 <- FLlst(list(a=1:10, b=10:20))
fll05 <- FLlst(c(1:10), c(10:20))
names(fll05) <- names(fll01)
names(fll01)
```

FLModel *Class FLModel for statistical models*

Description

The FLModel class provides a virtual class that developers of various statistical models can use to implement classes that allow those models to be tested, fitted and presented.

Slots in this class attempt to map all the usual outputs for a modelling exercise, together with the standard inputs. Input data is stored in slots created by each of those classes based on FLModel.See, for example FLSR for a class used for stock-recruitment models.

Various fitting algorithms, similar to those present in the basic R packages are currently available for FLModel, including fmle, nls-FLCore and glm.

Slots

name Name of the object. character.

desc Description of the object. character.

range Range. numeric.

fitted Estimated values for rec. FLQuant.

residuals Residuals obtained from the model fit. FLQuant.

model Model formula. formula.

gr Function returning the gradient of the likelihood. function.

logl Log-likelihood function. function.

initial Function returning initial parameter values for the optimizer, as an object of class FLPar. function.

params Estimated parameter values. FLPar.

logLik Value of the log-likelihood. logLik.

vcov Variance-covariance matrix. array.

hessian Hessian matrix obtained from the parameter fitting. array.

details extra information on the model fit procedure. list.

Extends

FLComp

Constructor

Constructor method for objects of class FLModel. This method is to be called by the constructor methods of classes extending FLModel.

An argument class instructs the constructor about the exact class of the returned object. Constructor methods for FLModel-based class should simply invoke this method with the appropriate class argument (See example below).

signature(object=formula) : First argument is a formula describing the model, to be placed in the model slot.

signature(object=missing) : Arguments, if given, are parsed and allocated by name to a iven slot. If none if provided, and empty FLModel object is returned.

signature(object=function) : A function returning a list with names equal to one or more of the slots in the class is called, and elements in that list are allocated by name. See SRModels for uses of this mechanism

signature(object=character) : A function with his name is called, as above.

Methods

AIC(stats) : Describe method

`signature(object=FLModel,k=numeric)`

AIC(stats) : Describe method

`signature(object=FLModel,k=missing)`

BIC(stats) : Describe method

`signature(object=missing)`

fmle(FLCore) : Describe method

`signature(object=missing,start=missing)`

nls(stats) : Describe method

`signature(formula=FLModel,data=missing,start=missing,control=missing,algorithm=missing,trace=m`

Author(s)

The FLR Team

See Also

AIC, BIC, fmle, nls, FLComp

Examples

```
# Normally, FLModel objects won't be created, as class lacks input slots
summary(FLModel(length~width*alpha))

# Objects of FLModel-based classes use their own constructor,
# which internally calls FLModel
fsr <- FLModel(rec~ssb*a, class='FLSR')
is(fsr)
summary(fsr)

# An example constructor method for an FLModel-based class
# create FLGrowth class with a single new slot, 'mass'
setClass('FLGrowth', representation("FLModel",
  mass='FLArray'))

# define creator method, based on FLModel()
setGeneric('FLGrowth', function(object, ...)
standardGeneric('FLGrowth'))
setMethod('FLGrowth', signature(object='ANY'),
  function(object, ...)
    FLModel(object, class='FLGrowth', ...))
```

FLPar *Class FLPar*

Description

The FLPar class is a class for storing the parameters of a model. It is based on the array class which can store Monte Carlo samples and the names of the relevant parameter vectors.

Methods for this class include subsetting and replacement as they exist for the FLQuant class. There are methods for extracting statistics of the sample (mean, median etc.) and for plotting the parameter samples.

Slots

.Data Describe slot. array.

units Units of measurement. character.

Extends

array

Constructor

The FLPar(object) constructor method allows simple creation of new FLPar with the methods described below.

signature(object=array) : Describe method

signature(object=missing) : Describe method

signature(object=vector) : Describe method

Methods

[(base) : Subsetting method to access the parameter values.

signature(x=FLPar,i=ANY,j=ANY,drop=missing)

[<-(base) : Replacement method for the parameter values.

signature(x=FLPar,i=ANY,j=ANY,value=missing)

as.data.frame(base) : Creates a data frame from the object.

signature(x=FLPar,row.names=ANY,optional=ANY)

densityplot(lattice) : applies the densityplot method from lattice

signature(x=formula,data=FLPar)

histogram(lattice) : applies the histogram method from lattice

signature(x=formula,data=FLPar)

iter(FLCore) : extracts the relevant iteration for the given parameter sample.

signature(object=missing)

iter<-(FLCore) : replacement method for the parameter iteration.

signature(object=missing,value=missing)

mean(base) : calculates the mean of the parameter samples.

signature(x=missing)

median(stats) : calculates the median of the parameter samples.

signature(x=missing,na.rm=missing)

plot(graphics) : default plot method for FLPar

signature(x=FLPar,y=missing)

splom(lattice) : applies the splom method from lattice

signature(x=FLPar,data=missing)

summary(base) : Summarises the FLPar object.

signature(object=missing)

units(base) : extracts the units of the parameters.

signature(x=missing)

units<-(base) : replacement method for the units of the object.

signature(x=FLPar,value=character)

var(stats) : calculates the variance of the parameter samples.

signature(x=missing,y=missing,na.rm=missing,use=missing)

Author(s)

The FLR Team

See Also

[, [<-, as.data.frame, densityplot, histogram, iter, iter<-, mean, median, plot, splom, summary, units,FLPar-method, units<-,FLPar,character-method, var

Examples

```
FLPar(rnorm(4), params=c('a','b','c','sigma2'))
```

FLQuant *FLQuant class for numerical data*

Description

The FLQuant class is a six-dimensional array designed to store most quantitative data used in fisheries and population modelling.

Details

The six dimensions are named. The name of the first dimension can be altered by the user from its default, quant. This could typically be age or length for data related to natural populations. The only name not accepted is 'cohort', as data structured along cohort should be stored using the FLCohort class instead. Other dimensions are always names as follows: year, for the calendar year of the datapoint; unit, for any kind of division of the population, e.g. by sex; season, for any temporal strata shorter than year; area, for any kind of spatial stratification; and iter, for replicates obtained through bootstrap, simulation or Bayesian analysis.

In addition, FLQuant objects contain a units attribute, of class character, intended to contain the units of measurement relevant to the data.

Slots

.Data A 6-D array for numeric data. array.

units Units of measurement. character.

Validity

Dimensions: Array must have 6 dimensions

Content: Array must be of class numeric

Dimnames: Dimensions 2 to 6 must be named "year", "unit", "season", "area" and "iter"

Constructor

The FLQuant method provides a flexible constructor for objects of the class. Inputs can be of class:

vector**:** A numeric vector will be placed along the year dimension by default.

matrix**:** A matrix will be placed along dimensions 1 and 2, unless otherwise specified by 'dim'. The matrix dimnames will be used unless overriden by 'dimnames'.

array: As above

missing: If no input is given, an empty FLQuant (NA) is returned, but dimensions and dimnames can still be specified.

Additional arguments to the constructor:

units: The units of measurement, a character string.

dim: The dimensions of the object, a numeric vector of length 6.

dimnames: A list object providing the dimnames of the array. Only those different from the default ones need to be specified.

quant: The name of the first dimension, if different from 'quant', as a character string.

Author(s)

The FLR Team

See Also

FLQuant

Examples

```
# creating a new FLQuant
flq <- FLQuant()
flq <- FLQuant(1:10, dim=c(2,5))
summary(flq)

# Vectors are used column first...
dim(FLQuant(1:10))
# ...while matrices go row first.
dim(FLQuant(matrix(1:10)))

FLQuant(matrix(rnorm(100), ncol=20))
```

```
FLQuant(array(rnorm(100), dim=c(5,2,1,1,1,10)))
FLQuant(array(rnorm(100), dim=c(5,2)), iter=10)

# working with FLQuant objects
flq <- FLQuant(rnorm(200), dimnames=list(age=1:5, year=2000:2008), units='diff')
summary(flq)

flq[1,]
flq[,1]
flq[1,1] <- 0

units(flq)
quant(flq)

plot(flq)
```

FLQuantPoint *Class FLQuantPoint*

Description

The FLQuantPoint class summarizes the contents of an FLQuant object with multiple iterations along its sixth dimension using a number of descriptive statistics.

Details

An object of this class has a set structure along its sixth dimension (*iter*), which will always be of length 5, and with dimnames *mean*, *median*, *var*, *uppq* and *lowq*. They refer, respectively, to the sample mean, sample median, variance, and lower (0.25) and upper (0.75) quantiles.

Objects of this class wil be typically created from an FLQuant. The various statistics are calculated along the *iter* dimension of the original FLQuant using apply.

Slots

.Data The main array holding the computed statistics. array.

units Units of measurement. character.

Accesors

mean,mean<-: 'mean' element on 6th dimension, arithmetic mean.

median,median<-: 'median' element on 6th dimension, median.

var,var<-: 'var' element on 6th dimension, variance.

lowq,lowq<-: 'lowq' element on 6th dimension, lower quantile (0.25 by default).

uppq,uppq<-: 'uppq' element on 6th dimension, upper quantile (0.75 by default).

Constructor

Inputs can be of class:

FLQuant**:** An FLQuant object with iters (i.e. dim[6] > 1)

Validity

iter: iter dimension is of length 5.

Dimnames: iter dimnames are 'mean', 'median', 'var', 'uppq' and'lowq'

Author(s)

The FLR Team

See Also

FLQuant

Examples

```
flq <- FLQuant(rnorm(2000), dim=c(10,20,1,1,1,200))
flqp <- FLQuantPoint(flq)
summary(flqp)
mean(flqp)
var(flqp)
rnorm(200, flqp)
```

FLQuantPoint-accesors *Method lowq*

Description

These are the accesor and replacement methods for the various elements stored in an FLQuantPoint object along the sixth dimension.

Generic function

lowq(x) lowq<-(x,value) mean(x) mean<-(x,value) median(x,na.rm) median<-(x,value) uppq(x) uppq<-(x,value) var(x,y,na.rm,use) var<-(x,value)

Methods

signature(x=FLQuantPoint) : Returns the given *iter*

signature(x=FLQuantPoint, value=FLQuant) : Replaces the given *iter* with the *value* FLQuant

Author(s)

The FLR Team

See Also

FLComp

Examples

```
flq <- FLQuant(rnorm(2000), dim=c(10,20,1,1,1,200))
flqp <- FLQuantPoint(flq)
mean(flqp)
mean(flqp) <- FLQuant(rnorm(200, 10, 3), dim=c(10,20))
```

FLQuants *Class FLQuants*

Description

FLQuants is a list of FLQuant objects. It is very similar to the standard list class. It implements a lock mechanism that, when turned on, does not allow the user to increase or decrease the object length. The elements of FLQuants must all be of class FLQuant.

Slots

.Data The data. list.

names Names of the list elements. character.

desc Description of the object. character.

lock Lock mechanism, if turned on the length of the list can not be modified by adding or removing elements. logical.

Accesors

Elements in the list can be accessed using '[' and '[['

Constructor

A constructor method exists for this class that can take named arguments for any of the list elements.

Author(s)

The FLR Team

See Also

*, Arith, as.data.frame, bubbles, catch<-, iter, model.frame, show, summary, xyplot, FLlst, list

FLQuantSums *Methods to compute sums, means and vars of FLQuant objects*

Description

This set of methods computes three different summaries (sum, mean and variance) of an FLQuant object along each of the six dimensions (quant, year, unit, season, area, or iter). Three methods (dimSums, dimMeans and dimVars) operate by default over the second to fifth dimensions (unit, season and area).

These methods simply encapsulate a call to apply with the corresponding dimension and function.

Sums are not calculated for the iter dimension, as it is used to store multiple replicates of a given array of values.

Methods to operate over the first dimension refer to it as the quant dimension, regardless of the actual name used in the object.

The output object will have length=1 on the selected dimension.

Generic function

quantSums(x), quantMeans(x), quantVars(x)

yearSums(x), yearMeans(x), yearVars(x)

unitSums(x), unitMeans(x), unitVars(x)

seasonSums(x), seasonMeans(x), seasonVars(x)

areaSums(x), areaMeans(x), areaVars(x)

iterMeans(x), iterVars(x)

dimSums(x), dimMeans(x), dimVars(x)

Method arguments

x : an object of a class for which this method has been defined.

na.rm : a logical indicating whether NAs should be deleted from the calculations. Defaults to TRUE.

dim : numeric, the dimensions over which dimSums, dimMeans or dimVars should operate. Defaults to c(1:2,6).

Methods

signature(x=FLQuant) : Computes a given summary statistic over a certain dimension of an FLQuant.

Author(s)

The FLR Team

See Also

FLQuant, sum, mean, var

Examples

```
flq <- FLQuant(rnorm(4000), dim=c(5,10,2,2,2,10), quant='age')
quantSums(flq)
quantMeans(flq)
yearSums(flq)
iterMeans(flq)
dim(quantSums(flq))
```

FLQuantTotals *Method quantTotals*

Description

These methods return an object of same dimensions as the input but with the sums along the first (yearTotals) or second dimension (quantTotals). Although the names might appear contradictory, it must be noted that what each method really returns are the totals over the selected dimension.

Generic function

quantTotals(x)

yearTotals(x)

Methods

signature(x=FLQuant) : Compute totals for an FLQuant dimension and replicate to the whole object

Author(s)

The FLR Team

See Also

FLComp

Examples

```
flq <- FLQuant(rlnorm(100), dim=c(10,10))
quantTotals(flq)
# See how the values obtained by yearSums are being replicated
yearSums(flq)
# Get the proportions by quant
flq / quantTotals(flq)
# or year
flq / yearTotals(flq)
```

FLSR *Class FLSR*

Description

Class for stock-recruitment models.

Details

A series of commonly-used stock-recruitment models are already available, including the corresponding likelihood functions and calculation of initial values. See SRModels for more details and the exact formulation implemented for each of them.

Slots

name Name of the object (character).

desc Description of the object (character).

range Range (numeric).

rec Recruitment series (FLQuant).

ssb Index of reproductive potential, e.g. SSB or egg oor egg production (FLQuant).

fitted Estimated values for rec (FLQuant).

residuals Residuals obtained from the model fit (FLArray).

covar Covariates for SR model (FLQuants).

model Model formula (formula).

gr Function returning the gradient of the likelihood (function).

logl Log-likelihood function (function).

initial Function returning initial parameter values for the optimizer (function).

params Estimated parameter values (FLPar).

logLik Value of the log-likelihood (logLik).

vcov Variance-covariance matrix (array).

details Extra information on the model fit procedure (list).

logerror Is the error on a log scale (logical).

distribution (factor).

hessian Resulting Hessian matrix from the fit (array).

Author(s)

The FLR Team

See Also

FLModel, FLComp

Examples

```
  # Create an empty FLSR object.
  sr1 <- FLSR()

  # Create an  FLSR object using the existing SR models.
  sr2 <- FLSR(model = 'ricker')
  sr2@model
  sr2@initial
  sr2@logl

  sr3 <- FLSR(model = 'bevholt')
  sr3@model
  sr3@initial
  sr3@logl

  # Create an FLSR using a function.
  mysr1 <- function(){
    model <- rec ~ a*ssb^b
    return(list(model = model))}

  sr4 <- FLSR(model = mysr1)

  # Create an FLSR using a function and check that it works.
  mysr2 <- function(){
    formula <- rec ~ a+ssb*b

    logl <- function(a, b, sigma, rec, ssb) sum(dnorm(rec,
      a + ssb*b, sqrt(sigma), TRUE))

   initial <- structure(function(rec, ssb) {
      a   <- mean(rec)
      b   <- 1
      sigma <- sqrt(var(rec))

      return(list(a=a, b=b, sigma=sigma))},
        lower = c(0, 1e-04, 1e-04), upper = rep(Inf, 3))

   return(list(model = formula, initial = initial, logl = logl))
  }

  ssb <- FLQuant(runif(10, 10000, 100000))
  rec <- 10000 + 2*ssb + rnorm(10,0,1)
  sr5 <- FLSR(model = mysr2, ssb = ssb, rec = rec)

  sr5.mle <- fmle(sr5)
  sr5.nls <- nls(sr5)

# NS Herring stock-recruitment dataset
data(nsher)

# already fitted with a Ricker SR model
summary(nsher)

plot(nsher)
```

```
# change model
model(nsher) <- bevholt()

# fit through MLE
nsher <- fmle(nsher)

plot(nsher)
```

FLStock *Class FLStock*

Description

A class for modelling a fish stock.

Details

The FLStock object contains a representation of a fish stock This includes information on removals (i.e. catches, landings and discards), maturity, natural mortality and the results of an analytical assessment (i.e. estimates of abundance and removal rates).

Slots

catch Total catch weight (FLQuant).

catch.n Catch numbers (FLQuant).

catch.wt Mean catch weights (FLQuant).

discards Total discards weight (FLQuant).

discards.n Discard numbers (FLQuant).

discards.wt Mean discard weights (FLQuant).

landings Total landings weight (FLQuant).

landings.n Landing numbers (FLQuant).

landings.wt Landing weights (FLQuant).

stock Total stock weight (FLQuant).

stock.n Stock numbers (FLQuant).

stock.wt Mean stock weights (FLQuant).

m Natural mortality (FLQuant).

mat Proportion mature (FLQuant).

harvest Harvest rate or fishing mortality. The units of the FLQuant should be set to 'harvest' or 'f' accordingly (FLQuant).

harvest.spwn Proportion of harvest/fishing mortality before spawning (FLQuant).

m.spwn Proportion of natural mortality before spawning (FLQuant).

name Name of the stock (character).

desc Description of stock (character).

range Named numeric vector containing the quant and year ranges, the plusgroup and the quant range that the average fishing mortality is calculated over (numeric).

Validity

Dimensions All FLQuant slots must have iters equal to 1 or 'n'.

Iters The dimname for iter[1] should be '1'.

Dimnames The name of the quant dimension must be the same for all FLQuant slots.

Totals The length of the quant dimension for the totals slots (catch, landings and discards) must be equal to 1.

Accesors

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 stored in any slot.

Author(s)

The FLR Team

See Also

[, [<-, as.FLSR, catch, catch<-, catch.n, catch.n<-, catch.wt, catch.wt<-, coerce, computeCatch, computeDiscards, computeLandings, discards, discards<-, discards.n, discards.n<-, discards.wt, discards.wt<-, harvest, harvest<-, harvest.spwn, landings, landings<-, landings.n, landings.n<-, landings.wt, landings.wt<-, m, m<-, mat, m.spwn, plot, ssb, ssbpurec, stock, stock.n, stock.wt, trim, FLComp

Examples

```
data(ple4)

# get the landings slot
landings(ple4) #get the landings slot
# assign values to the landings slot
landings(ple4) <- apply(landings.n(ple4)*landings.wt(ple4),2,sum)

discards(ple4) <- computeDiscards(ple4)

# set the units of the harvest slot of an FLStock object
harvest(ple4) <- 'f'

catch(ple4) <- computeCatch(ple4)
catch(ple4) <- computeCatch(ple4, slot="all")

ple4[,1] # subset the FLStock
# trim the FLStock
```

```
trim(ple4, age=2:6, year=1980:1990)

# calculate SSB
ssb(ple4)
# calculate SSB per recruit
ssbpurec(ple4)

# coerce an FLStock to an FLBiol
biol <- as(ple4, "FLBiol")
# initialise an FLSR object from an FLStock
flsr <- as.FLSR(ple4)
```

FLStockLen *Class FLStockLen*

Description

A class for modelling a length structured fish stock.

Details

The FLStockLen object contains a length based representation of a fish stock This includes information on removals (i.e. catches, landings and discards), maturity, natural mortality and the results of an analytical assessment (i.e. estimates of abundance and removal rates).

Slots

halfwidth The middle of the length bins (numeric).

catch Total catch weight (FLQuant).

catch.n Catch numbers (FLQuant).

catch.wt Mean catch weights (FLQuant).

discards Total discards weight (FLQuant).

discards.n Discard numbers (FLQuant).

discards.wt Mean discard weights (FLQuant).

landings Total landings weight (FLQuant).

landings.n Landing numbers (FLQuant).

landings.wt Landing weights (FLQuant).

stock Total stock weight (FLQuant).

stock.n Stock numbers (FLQuant).

stock.wt Mean stock weights (FLQuant).

m Natural mortality (FLQuant).

mat Proportion mature (FLQuant).

harvest Harvest rate or fishing mortality. The units of the FLQuant should be set to 'harvest' or 'f' accordingly (FLQuant).

harvest.spwn Proportion of harvest/fishing mortality before spawning (FLQuant).

m.spwn Proportion of natural mortality before spawning (FLQuant).

name Name of the stock (`character`).

desc Description of stock (`character`).

range Named numeric vector containing the quant and year ranges, the plusgroup and the quant range that the average fishing mortality is calculated over (`numeric`).

Accesors

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 stored in any slot.

Validity

Dimensions All FLQuant slots must have iters equal to 1 or 'n'.

Iters The dimname for iter[1] should be '1'.

Dimnames The name of the quant dimension must be the same for all FLQuant slots.

Totals The length of the quant dimension for the totals slots (catch, landings and discards) must be equal to 1.

Author(s)

The FLR Team

See Also

[, [<-, as.FLSR, computeCatch, computeDiscards, computeLandings, plot, ssb, ssbpurec, trim, FLComp

---

`FLStocks` *Class FLStocks*

Description

`FLStocks` is a class that extends `list` through `FLlst` but implements a set of features that give a little bit more structure to list objects. The elements of `FLStocks` must all be of class `FLStock`. It implements a lock mechanism that, when turned on, does not allow the user to increase or decrease the object length.

Slots

.Data The data. `list`.

names Names of the list elements. `character`.

desc Description of the object. `character`.

lock Lock mechanism, if turned on the length of the list can not be modified by adding or removing elements. `logical`.

Extends

FLlst list vector

Constructor

The `FLStocks(object)` constructor method allows simple creation of new FLStocks with the methods described below.

signature(object=ANY) : Returns an FLStocks object with the given named elements

signature(object=missing) : Returns an FLStocks object

signature(object=list) : Returns an FLStocks object with the provided list as its core

Methods

plot(graphics): Default plot for FLStocks. Different lines for each stock object are shown in panel panels, corresponding to *rec*, *ssb*, *catch* and *harvest*. A *key* argument turns off (FALSE, default) or on (TRUE) a figure key. A new key can also be provided (see *key* entry in `xyplot`.

`signature(x=FLStocks,y=missing)`

Author(s)

The FLR Team

See Also

plot, FLlst, list

Examples

```
data(ple4)
fls <- FLStocks(sa=ple4, sb=window(ple4, end=1980))
summary(fls)
```

`fmle` *Method fmle*

Description

The `fmle` method fits the model specified in an `FLModel` object using MLE by minimizing the negative of the log-likelihood function, in the `logl` slot, through calls to the `optim` minimizaton routine.

For a given model and log-likelihood function, the `fmle` method will use the `optim` function in R to calculate the parameter vector which maximises the log- likelihood (and, hence, the likelihood function) and is as such the optimum parameter value for the given problem and data.

Be advised that for non-informative of conflicting data the maximum likelihood estimate can be dependent on the initial starting value and if we begin the optimiser with a poor initial estimate it may converge falsely. Always try multiple start points and be assured that you ave found the true MLE.

Generic function

fmle(object,start)

Methods

signature(object=ANY,start=missing) : Generic method.

signature(object=FLModel,start=ANY) : Input object of class `FLModel` contains input data, logl function and function to provide initial values.

signature(object=FLModel,start=FLPar) : Input object of class `FLModel` contains input data and logl function, but fitting is started from parameter estimates in the `FLPar` object provided.

Author(s)

The FLR Team

See Also

FLComp

Examples

```
# use an example FLModel object
data(nsher)

summary(nsher)

# inspect the logl function
logl(nsher)

# and the function providing initial values to the optimizer
initial(nsher)

# lower and upper limits for the parameters are set, and used if method
# 'L-BFGS-B' is used in the call to optim, as is default in fmle
```

```
lower(nsher)
upper(nsher)

# fit it with fmle
nsher <- fmle(nsher)

# fixed values can be chosen for any parameter
nsher_fixed_a <- fmle(nsher, fixed=list(a=125))

# and results compared, for example using AIC
AIC(nsher)
AIC(nsher_fixed_a)

## Not run:
# an initial run with one optimization method, e.g. 'SANN'
nsher_one <- fmle(nsher, method='SANN')

# can then be used as starting value for other runs
# This might fail if
nsher_two <- fmle(nsher_one, start=params(nsher_one), method='L-BFGS-B')

## End(Not run)
```

`harvest` *Harvest calculations for FLBiol*

Description

Calculates the fishing mortality (F), based on abundance changes by year and age, and the difference between total mortality (Z) and natural mortality (M), for an object of class `FLBiol`.

Generic function

harvest(object)

Methods

signature(object=FLBiol) : Uses the method above to calculate an approximation of harvest rate

Author(s)

The FLR Team

See Also

FLComp, FLBiol

Examples

```
data(ple4)
flb <- as(ple4, 'FLBiol')
harvest(flb)
```

IOfunctions *Input/Output of FLR objects*

Description

These functions read and save FLR objects of class FLStock, FLIndex and FLQuant to and from various datafile formats commonly used in fisheries work and stock assessment.

Usage

```
readFLStock(file, type = "VPA", name,
  desc = paste("Imported from a", type, "file. (", file, "). ", date()),
  m = 0.2, quant = "age", quiet = TRUE, no.discards = FALSE, sep="")
readVPAFile(file, sep = "", units = "NA", quiet = TRUE)
readFLIndex(file, type = "VPA", index.names, descs,
  desc =  paste("Imported from ", type, " file '", file, "'", sep = ""))
readFLIndices(file, file2, type = "VPA", index.names, descs,
  desc = paste("Imported from ", type, " file '", file, "'", sep = ""),
  na.strings = "NA", sep="")
writeFLStock(FLStock, output.file=FLStock@name, type = "VPA")
```

Arguments

file, file2	name of file containing data in correct format.
output.file	directory and base filename where to place 'Lowestoft VPA Suite FOrmat' files
type	this can either be "VPA" or "adapt" for 'Lowestoft' or 'ICCAT' format
name	name for object created
index.names	names for individual objects in FLIndices
desc	description for object created
descs	descriptions for individual objects in FLIndices
m	natural mortality, default = 0.2, only used for 'ICCAT Adapt Format'
quant	name for quant dimension default is "age"
quiet	logical, suppress chit-chat
sep	character separating columns of data
na.strings	string used to represent NA values in the input files
no.discards	should discards be assumed to be zero?
units	units of measurement to be stored in the units attribute
FLStock	FLStock object to be saved

Details

These functions are used for reading and writing stock and CPUE data used to conduct stock assessment. A number of data input formats are currently supported. These include the 'Lowestoft VPA Suite file format' which comprises a number of flat ascii data files for catch numbers at age, catch weights at age, maturity, etc. and the 'ICCAT Adapt Format'

For the 'Lowestoft VPA Suite file format' each input file contains header information specifying the dimensions of the data matrix which may be comma, space or tab delimited. Any comments in

the file must be prefixed with a '\#'. An index file gives the names of the individual data files to be read in. This is the file that should be passed to `readFLStock`. A single file can be read by `readVPA` into an `FLQuant`. «««< HEAD Further information on the VPA file format can be found in the XSA manual. ======= »»»> master

The 'Adapt file format' comprises a single file for input containing both the biological parameters, catches and catch per unit effort data.

If information on discards numbers at age and discards weights at age are available and these files are specified in the index file then they will be read into the `FLStock` object otherwise the discards slots in the `FLStock` object will remain empty.

For reading CPUE data into an FLindex the file containing the CPUE data should be passed to the `readFLIndex` function if only one CPUE series is given or else to `readFLIndices` if multiple series are given.

Confusingly, the file giving the names of the individual `FLStock` input files is often called the index file. It is different to the file containing the CPUE data used for 'tuning' an assessment which is also sometimes called the index file.

`writeFLStock` creates a set of files in 'Lowestoft File format'. This function takes `output.file` as its argument, which is the base filename from which the output files will be named according to their content. A directory can be included in this argument (e.g. via `file.path`) to specify where the files should be written.

Value

An object of class `FLQuant`, `FLStock` or `FLIndex` depending on the function.

Author(s)

The FLR Team

References

Darby, C.D. and Flatman, F. Virtual Population Analysis: version 3.1 (Windows/DOS) user guide. Information technology series, No 1. CEFAS, Lowestoft, UK

Porch, C. E. 1997. A user's manual for VPA-2BOX Version 2.0. National Marine Fisheries Service, Miami, USA.

See Also

FLQuant, FLStock, FLIndex.

Examples

```
## Not run:
path  <- getwd()

## reads a set of 'Lowestoft File Format' for a stock and creates an FLStock object
ple4 <- readFLStock(paste(path, "pleindex.txt", sep=""))

## reads a single file in 'Lowestoft File Format' and creates an FLQuant
ple4.catch.n <- readVPAFile(paste(path, "plecanum.txt", sep=""))

## reads a set of tuning data in 'Lowestoft File Format' and creates an FLIndices object
ple4.indices <- readFLIndices(paste(path, "plecpue.txt", sep=""))
```

```
## reads a single index from a tuning file in 'Lowestoft File Format' and creates an FLIndex object
ple4.index <- readFLIndex(paste(path, "plecpue1.txt", sep=""))

## writes an FLStock to 'Lowestoft File Format' in the current working directory
writeFLStock(ple4,output.file=file.path(getwd(),"Ple4"))

## End(Not run)
```

`is.FL` *Methods to determine the class of a given object*

Description

These methods return `TRUE` if the given object is of the corresponding class, and `FALSE` otherwise.

These methods should be substituted by calls to `is` and will very likely be deprecated in future releases.

Generic function

is.FLQuants(object)

is.FLBiols(object)

is.FLIndices(object)

is.FLStocks(object)

Methods

signature(object=ANY) : Describe method

Author(s)

The FLR Team

See Also

is

Examples

```
# This call ...
is.FLQuant(FLQuant())
# ... should be substituted by
is(FLQuant(), 'FLQuant')
```

`iter` *Select or modify iterations of an FLR object*

Description

To extract or modify a subset of the iterations contained in an FLR object, the `iter` and `iter<-` methods can be used.

In complex with various `FLQuant` slots, the `iter` method checks whether individual slots contain more than one iteration, i.e. `dims(object)[6] > 1`. If a particular slot contains a single iteration, that is returned, otherwise the chosen one is selected. This is in contrast with the subset operator `[`, which does not carry out this check.

For objects of class `FLModel`, iters are extracted for slots of classes `FLQuant`, `FLCohort` and `FLPar`.

Generic function

iter(object) iter<-(object,value)

Methods

signature(object=FLQuant) : Describe method

signature(object=FLComp) : Describe method

signature(object=FLQuants) : Describe method

signature(object=FLPar) : Describe method

Author(s)

The FLR Team

See Also

FLComp, FLQuant

Examples

```
flq <- FLQuant(rnorm(800), dim=c(4,10,2), iter=10)
iter(flq, 2)

fls <- FLStock(catch.n=flq, m=FLQuant(0.2, dim=c(4,10,2)))
fls2 <- iter(fls, 2)
summary(fls2)
```

iters *Method iters*

Description

Displays all the iterations of an FLQuant object.

Generic function

iters(object)

Methods

signature(object=FLQuant) : Displays all the iterations of the object

Author(s)

The FLR Team

See Also

FLComp

Examples

```
a <- FLQuant(1:24, dim = c(2,3,1,1,1,4))
a
iters(a)
```

jacknife *Jacknife resampling*

Description

The jacknife method sets up objects ready for jacknifing, i.e. to systematically recompute a given statistic leaving out one observation at a time. From this new set of "observations" for the statistic an estimate for the bias can be calculated as well as an estimate for the variance of the statistic.

Input objects cannot have length > 1 along the iter dimension, and the resulting object will have as many iterations as elements in the original object.

Generic function

jacknife(object, ...)

Methods

signature(object=FLQuant) : Returns an FLQuant with iter=length(object), where in each iteration one element has been sequantally converted to NA.

Author(s)

The FLR Team

See Also

FLQuant

Examples

```
flq <- FLQuant(1:8)
iters(jacknife(flq))
```

---

lapply *Method lapply*

Description

lapply returns a list of the same length as X, each element of which is the result of applying FUN to the corresponding element of X.

Generic function

lapply(X,FUN)

Methods

signature(X=FLlst,FUN=missing) : lapply returns a list or FLlst of the same length as X, each element of which is the result of applying FUN to the corresponding element of X of class FLlst.

Author(s)

The FLR Team

See Also

FLComp

Examples

```
# On an FLQuants object
flqs <- FLQuants(a=FLQuant(1:10), b=FLQuant(1:20))
# lapply could return another FLQuants object
lapply(flqs, yearSums)
# or a simple list, depending on the function  being called
lapply(flqs, dim)
```

`lattice` *Lattice plots*

Description

Implementation of Trellis graphics in FLR. Plot methods in the `lattice` package are available for object of class `FLQuant`, `FLQuants` or those derive from `FLComp`.

See the help page in `lattice` for a full description of each plot method and of all possible arguments.

Plot methods from lattice are called by passing a data.frame obtained by converting the FLR objects using as.data.frame. For details on this transformation, see as.data.frame-FLCore.

Generic function

barchart(x, data, ...)

bwplot(x, data, ...)

densityplot(x, data, ...)

dotplot(x, data, ...)

histogram(x, data, ...)

stripplot(x, data, ...)

xyplot(x, data, ...)

Methods

signature(x=formula, data=FLQuant) : Use the lattice functionality for objects of class FLQuant

signature(x=formula, data=FLQuants) : Use the lattice functionality for objects of class FLQuants

signature(x=formula, data=FLComp) : Use the lattice functionality for objects of class FLComp

signature(x=formula, data=FLPar) : Use the lattice functionality for objects of class FLPar

Author(s)

The FLR Team

See Also

xyplot, barchart, bwplot, densityplot, dotplot, histogram, stripplot

Examples

```
data(ple4)
# xyplot on FLQuant
xyplot(data~year|age, catch.n(ple4)[, 1:20])

xyplot(data~year|as.factor(age), catch.n(ple4)[, 1:20], type='b', pch=19, cex=0.5)

# bwplot on FLQuant with iter
flq <- rnorm(100, catch.n(ple4)[, 1:20], catch.n(ple4)[,1:20])
bwplot(data~year|as.factor(age), flq)
```

```
# now with same style modifications
bwplot(data~year|as.factor(age), flq, scales=list(relation='free',
  x=list(at=seq(1, 20, by=5), labels=dimnames(catch.n(ple4)[,1:20])$year[seq(1, 20,
  by=5)])), cex=0.5, strip = strip.custom(strip.names = TRUE, strip.levels = TRUE,
  var.name='age'))
```

`leslie` *Method for calculating Leslie matrix dynamics of an FLBiol object*

Description

For an `FLBiol` object with the natural mortality-at-age, fecundity and spwn data present in the object.

Usage

```
leslie(object, ...)
```

Arguments

`object`	An object of type `FLBiol`.
`...`	Extra arguments accepted by each implementation.

Details

Usual Leslie matrix type dynamics for a `FLBiol` object.

Value

An object of class `FLBiol`.

Author(s)

FLR Team

References

Leslie, P.H. (1945) The use of matrices in certain population mathematics. Biometrika,33(3), 183-212.

Leslie, P.H. (1948) Some further notes on the use of matrices in population mathematics. Biometrika, 35(3-4), 213-245.

See Also

`FLBiol`

Examples

```
data(ple4.biol)
ple4.l <- leslie(ple4.biol,plusgroup=FALSE)
```

limits *Methods upper and lower*

Description

Accesor and replacement methods for the lower and upper attributes of objects of class FLModel. These are stored as part of the structure inside the initial slot. This slot contains a function to be used to provide initial values to any of the fitting method (e.g. fmle).

The values in lower and upper are only used if the method selected for optim is able to make use of them, like for example "L-BFGS-B", which is the default for fmle.

The exact location of this information could be changed (i.e. a separate slot might be created), so code accessing it is encouraged to use these accesor methods.

Generic function

upper(object) upper<-(object, value) lower(object) lower<-(object, value)

Methods

signature(object=FLModel) : Describe method

signature(object=FLModel, value=numeric) : Describe method

Author(s)

The FLR Team

See Also

FLModel

Examples

```
data(nsher)
lower(nsher)
upper(nsher)
```

lowess *Method lowess*

Description

LOWESS smoother based on locally-weighted polynomial regression for objects of class FLSR. The model fitted is of the form rec(x)~ssb(x). Returns an object of class FLQuants with elements named ssb and rec.

Generic function

lowess(x, y, f=2/3, iter=3, delta=0.01 * diff(range(xy\$x[o])))

Methods

signature(x=FLSR,y=missing) : lowess(x, y, f=2/3, iter=3, delta=0.01 * diff(range(ssb(x))))

Author(s)

The FLR Team

See Also

lowess, FLSR

Examples

```
# use the North Sea herring SR dataset
data(nsher)

# fitting a rec ~ ssb lowess
nshlos <- lowess(nsher)
```

---

mcf *Method mcf*

Description

This method makes FLQuants compatible with respect to their dimensionality. Hence, the FLQuants in the returned object all heve the same dimensions, padded with NAs if necessary

Generic function

mcf(object)

Methods

signature(object=FLComp) : All FLQuants in an FLComp object are made compatible with respect to their dimensionality

signature(object=list) : All FLQuants in an list are made compatible with respect to their dimensionality

Author(s)

The FLR Team

See Also

FLComp

Examples

```
fla <- FLQuant(rnorm(20), dim=c(2,10))
flb <- FLQuant(rnorm(45), dim=c(3,15))
fls <- FLQuants(a=fla, b=flb)
flc <- mcf(fls)
lapply(flc, dim)
```

mean *Method mean*

Description

Calculates the arithmetic mean. Can be used directly on an object or with apply etc.

Generic function

mean(x)

Methods

signature(x=FLPar) : Returns the mean of x

signature(x=FLQuantPoint) : Returns the mean of x

Author(s)

The FLR Team

See Also

median apply

Examples

```
flp <- FLPar(rnorm(80), params=c('a', 'b'), iter=1:40)
mean(flp)
```

mean.lifespan *Method for calculating mean lifespan, given the natural mortality*

Description

For an FLBiol object with the natural mortality-at-age present in the object.

Usage

```
mean.lifespan(x, ...)
```

Arguments

x	An object of type FLBiol.
...	Extra arguments accepted by each implementation.

Details

Using actuarial definitions for the expected life-span of a given species, for a given survival rate-at-age (natural mortality), we can compute the expected life-span, ℓ_x, of a species, from a given reference age x, using the following equation:

$$\ell_x = \sum_{t=1}^{\infty} \exp\left(-\sum_{i=x}^{x+t} M_i\right)$$

.

The method accepts objects of class `FLBiol` of any particular dimension. If the object has a seasonal structure to the population dynamics, then we sum over all seasons to get the yearly survival rate.

Value

An object of class `FLQuant` whose first and second dimension is of length one.

Author(s)

FLR Team

See Also

`FLBiol`

Examples

```
## Not run:
data(ple4biol)
lfs.ple4 <- mean.lifespan(ple4,ref.age=1)

## End(Not run)
```

---

median *Method median*

Description

Calculates the median.

Generic function

median(x,na.rm)

Methods

- signature(x=FLQuantPoint,na.rm=missing) :Returns the median of x, see `link`median,FLQuantPoint-method.
- signature(x=FLPar,na.rm=missing) :Returns the median of x along the *iter* dimension.

Author(s)

The FLR Team

See Also

median, apply

Examples

```
flp <- FLPar(rnorm(80), params=c('a', 'b'), iter=1:40)
iterMedians(flp)
```

---

mergeFL *Merging FLStock objects*

Description

Two FLStock object can be *merged* using this method or a plus sign. Catch slots are added, and weight slots are averaged, weighted by the relative catches. No meaningful calculation is currently done for harvest, harvest.spwn, m, and m.spwn.

Methods

signature(e1=FLStock, e2=FLStock) : Adds two FLStock objects

Author(s)

The FLR Team

See Also

FLStock

---

model.frame *Method model.frame*

Description

`model.frame` returns a `data.frame` with the variables in a wide format, to be used by a `formula` in any model method.

Generic function

model.frame(formula)

Methods

signature(formula=FLlst) : Returns a wide data.frame

signature(formula=FLComp) : Returns a wide data.frame

Author(s)

The FLR Team

See Also

model.frame, FLQuants, FLlst

Examples

```
data(ple4)
flqs <- FLQuants(stock=stock.n(ple4), catch=catch.n(ple4))
fmf <- model.frame(flqs)
head(fmf)
```

`names` *Method names*

Description

The `names` method returns the names of the dimnames of an object. For some classes, the names attribute can be modified directly using names<-.

Generic function

names(x) names<-(x, value)

Methods

signature(x=FLQuant) : Returns the names of the dimnames of x

signature(x=FLPar) : Returns the names of the dimnames of x

signature(x=FLlst) : Returns the names of the elements of x

Author(s)

The FLR Team

See Also

names

Examples

```
data(ple4)
names(catch.n(ple4))
```

nls *Method nls*

Description

For a given forumla (describing a model) and data this method applies the simple non- linear least squares algorithm - this calculates the parameters that minimisethe sum of squares difference between the observed (data) and predicted (model) values.

The algorithm can be sensitive to the initial values of the problem so do try different start points and check they converge to the same estimates.

Generic function

nls(formula,data,start,control,algorithm,trace,subset,weights,na.action,model,lower,upper)

Methods

signature(formula=FLModel,data=missing,start=missing,control=missing,algorithm=missing,trace=missing,sub
Applies non-linear sum of squares to the model and data in the input FLModel object.

Author(s)

The FLR Team

See Also

FLComp

Examples

```
# An example FLSR (FLModel) object
data(nsher)

#set bevholt model
model(nsher) <- bevholt

# fit through nls
nsher <- nls(nsher)

summary(nsher)
```

plot *Method plot*

Description

Standard plot methods for every FLR class. FLR plot methods are based on lattice, and attempt to show a general view of the object contents.

Users are encouraged to write their own plotting code make use of the overloaded lattice methods, for example xyplot or bwplot. See also lattice-FLCore.

Generic function

plot(x,y)

Methods

signature(x=FLQuant,y=missing) : Plot of an *FLQuant* conditioned on all dimension of length > 1.

signature(x=FLQuantPoint,y=missing) : Box and whiskers plot of the yearly time series, conditioned on all dimension of length > 1.

signature(x=FLPar,y=missing) : Densityplot per parameter.

signature(x=FLStock,y=missing) : Times series of catch and landings, recruitment, harvest and spawning stock biomass.

signature(x=FLStocks,y=missing) : Times series of catch, recruitment, harvest and spawning stock biomass.

signature(x=FLBiol,y=missing) : Time series of SSB and recruitment.

signature(x=FLCohort,y=missing) : Plot of an *FLQuant* conditioned on all dimension of length > 1.

signature(x=FLIndex,y=missing) : Either a time series of the standardised index by quant, type='ts', or and splom plot of a log-linear regression between quants, type='splom'.

signature(x=FLSR,y=missing) : A six-panelled plot showing the model fit, residuals by year, AR(1) residuals, residuals by SSB, residuals by estimated recruits and a normal Q-Q plot.

Author(s)

The FLR Team

See Also

plot

Examples

```
data(ple4)
data(ple4.biol)

# FLQuant
plot(catch.n(ple4)[, 1:20])
plot(catch.n(ple4)[, 1:20], type='b', pch=19, cex=0.5)

# FLStock
plot(ple4)

# FLBiol
plot(ple4.biol)
```

predict *Method predict*

Description

predict returns predicted values according to the parameter values and model formula in an FLModel object. If no extra input is given, *predict* will use the input values contained in the relevant slots. If any extra named argument is provided, this is used instead and the correspodning predicted values are returned.

Generic function

predict(object, ...)

Methods

signature(object=FLModel) : Calculates predicted values according to the fitted model

Author(s)

The FLR Team

See Also

FLComp

Examples

```
# nsher FLSR dataset
data(nsher)

# predict with no extra arguments returns the values
# predicted during model fitting
predict(nsher)

# which can also be extracted from the 'fitted' slot
fitted(nsher)

# a different ssb vector can be provided
predict(nsher, ssb=FLQuant(seq(10, 150, by=5)))
```

predictModel *A class for model prediction*

Description

Object of the predictModel class are used in various FLR classes to allow flexible modelling of the dynamics of different biological and technological processes.

Details

The dependency of life history processes, such as maturity and fecundity, to biological and environmental factors, can be represented in objects of this class via a simple model (represented by a 'formula') and the corresponding paramaters ('FLPar') and inputs ('FLQuants').

Slots

.Data Inputs to the model not found in enclosing class (`FLQuants`).

model Model representation (`formula`).

params Model paramaters (`FLPar`).

Validity

VALIDITY Neque porro quisquam est qui dolorem ipsum.

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

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.

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

`FLQuants` `FLPar` `FLBiol`

print *Method print*

Description

print prints its argument and returns it invisibly (via invisible(x)).

Generic function

print(x)

Methods

signature(x=FLQuant) : Prints the cntent of the object.

Author(s)

The FLR Team

See Also

FLComp

Examples

```
a <- FLQuant(1:6, dim = c(2,3))
for(i in 1:3) print(a[,1:i])
for(i in 1:3) a[,1:i]
```

propagate *Extend an FLQuant along the iter dimension*

Description

FLR objects with a single iteration (length of 1 in the sixth dimension) can be extended using the propagate method. The type argument selects whether the new iterations are filled with the content of the first iteration (type='all') or left empty (type='first').

For objects of class FLPar, propagate will extend the object along the last dimension, iter. The fill.iter argument defaults to FALSE, in contrast with FLQuant. Objects do not need to have iter=1 to be extended, but only if fill.iter=FALSE.

Generic function

propagate(object)

Arguments

object : An FLR object to extend.

fill.iter : Copy the first iteration along the new ones ('TRUE'), the default, or leave them empty ('FALSE').

iter : Number of iterations.

Methods

signature(object=FLQuant) : Extends object along the `iter` dimension

signature(object=FLPar) : Extends object along the `iter` dimension

signature(object=FLComp) : Extends all `FLQuant` slots in the object along the `iter` dimension

signature(object=FLPar) : Extends object along the `iter` dimension

Author(s)

The FLR Team

See Also

FLComp

Examples

```
flq <- FLQuant(rnorm(50), dim=c(5,10))
propagate(flq, 10)
# Look at the %NA in summary
summary(propagate(flq, 10, fill.iter=FALSE))

flp <- FLPar(1:10, params=letters[1:10])
propagate(flp, 10)
propagate(flp, 10, fill.iter=TRUE)

flp <- FLPar(1:15, params=letters[1:5], iter=3)
propagate(flp, 10)
```

---

pv *Population variability*

Description

The pv method computes the population variability (*pv*) of an `FLQuant` object.

Generic function

pv(object)

Methods

signature(object=FLQuant) : Describe method

Author(s)

The FLR Team

References

Heath, J.P. 2006. Quantifying temporal variability in population abundances. *Oikos* **115 (3)**: 573–581.

See Also

FLComp

Examples

```
flq <- FLQuant(rnorm(40), dim=c(1,40))
pv(flq)

data(ple4)
pv(stock(ple4))
```

---

qapply *Method qapply*

Description

Returns a `list` or `FLlst` containing values obtained by applying a function to margins for each FLQuant in a composite FLR object.

Generic function

qapply(X,FUN)

Methods

signature(X=FLComp,FUN=function) : `FUN` is typically a function name to be searched for from the environment of the call to `qapply`. Additional arguments to the function are specified after the function name.

`qapply` enables functions to be applied easily to all `FLQuants` of a composite object rather than repeating the code for each one separately. In the example below the `apply` function is nested inside `qapply` to calculate averages across various dimensions for each `FLQuant` in an `FLStock` object.

Author(s)

The FLR Team

See Also

FLComp apply

Examples

```
data(ple4)

# returns a list containing the max value for each quant
qapply(ple4, max)

# returns a FLStock of means across all dimensions except year
qapply(ple4, apply, 2, mean, na.rm=TRUE)

# returns an FLStock of max values across all dimensions except year and age
qapply(ple4, apply, c(1,2), max)
```

---

quant *Method quant*

Description

Function to get or set the name of first dimension (quant) in an object of class `FLQuant` or `FLCohort`.

Generic function

quant(object) quant<-(object,value)

Methods

signature(object=FLQuant) : Get the name of the first dimension of an `FLQuant` object.

signature(object=FLCohort) : Get the name of the first dimension of an `FLCohort` object.

signature(object=FLQuant,value=ANY) : Set the name of the first dimension of an `FLQuant` object.

Author(s)

The FLR Team

See Also

FLQuant, FLCohort

Examples

```
# quant is 'quant' by default
quant(FLQuant())

flq <- FLQuant(rnorm(80), dim=c(4,20), quant='age')

quant(flq)

quant(flq) <- 'length'

summary(flq)
```

quantile *Method quantile*

Description

Quantiles for FLQuant objects can be obtained with this method. Default quantiles returned are seq(0, 1, 0.25), but they can be specified using the probs argument. The returned FLQuant object uses the sixth dimension (*iter*) to store the requested quantiles, with appropriate dimnames.

For objects of class FLQuantPoint, quantile is merely an accessor for two elements of the sixth dimension, lowq and uppq. You could use the lowq and uppq methods instead.

Generic function

quantile(x, ...)

Methods

signature(x=FLQuant) : Describe method

signature(x=FLQuantPoint) : Describe method

Author(s)

The FLR Team

See Also

quantile, FLQuant, FLQuantPoint

Examples

```
# Normally distributed FLQuant, with log-normal random mean and fixed sd of 20
flq <- rnorm(100, FLQuant(rlnorm(20), dim=c(2,10)), 20)

# obtains all standard quantiles (0, 0.25, 0.5, 0.75 and 1)
quantile(flq)
# select one of them
quantile(flq)[,,,,,1]
# calculates the 0.05 quantile only
quantile(flq, 0.05)

# creates an FLQuantPoint from previous FLQuant
flp <- FLQuantPoint(flq)
# return each of the two quantiles (025 and 0.75)
quantile(flp, 0.25)
quantile(flp, 0.75)
```

r *Intrinsic rate of increase from an FLBiol object*

Description

For an FLBiol object with the mortality-at-age, fecundity and spwn data present in the object slots, this method calculates the intrinsic rate of increase r for the given population.

It does this using two methods:

(1) Solving the Euler-Lotka equation.

(2) Calculating the logarithm of the real part of the largest/lead eigenvalue of the Leslie transition matrix.

These two methods are not identical but do give similar answers for the same data and parameters

Usage

```
r(object, ...)
```

Arguments

object	An object of type FLBiol.
...	Extra arguments accepted by each implementation.

Details

To chose the method used to estimate r (Euler-Lotka or Leslie matrix) we supply either 'el' or 'leslie' as the 'method' argument (see below). To calculate r along years or cohorts by supply either 'year' or 'cohort' as the 'by' argument (see below).

The method can handle Monte Carlo samples (i.e. with iterations) in either the fec, m or both slots required to calculate r and the conversion is done internally so that we obtain an FLQuant of the correct dimensions.

Value

An object of class FLQuant.

Author(s)

FLR Team

See Also

FLBiol

Examples

```
## Not run:

# call in the NS herring stock and biol objects

data(nsher.biol)
data(nsher)

# calculate the gradient at the origin from the spawning stock numbers to the recruits

tmp <- nsher.biol
n(tmp) <- stock.n(nsher)
m(tmp) <- harvest(nsher)+m(nsher.biol)

# use nls to calculate gradient of (recruits/spawning stock numbers)
# assuming log-normal errors

dfx <- data.frame(rec=as.vector(n(tmp)[1,]),ssn=as.vector(ssn(tmp)))
res <- nls(log(rec)~log(a)+log(ssn),dfx,start=list(a=mean(n(tmp)[1,]/ssn(tmp))))

# use this value of recruits per spawner times maturity as the birth function for
# the Leslie transition matrix/Euler-Lotka equation

alpha <- coef(res)[[1]]
fec(tmp)[] <- fec(tmp)[] * alpha

# calculate r assuming only natural mortality and by year
# using both Euler-Lotka (el) and Lesie matrix (leslie) method

m(tmp) <- m(nsh.biol)
r.nsh.el <- r(tmp,by='year',method='el')
r.nsh.lm <- r(tmp,by='year',method='leslie')

## End(Not run)
```

---

range *Method range*

Description

Extraction and modification of the *range* slot from objects of any class inheriting from FLComp.

Generic function

range(x, i) range<-(x, i, value)

Methods

signature(x=FLComp, i=missing) : Returns the *range* slot.

signature(x=FLComp, i=character) : Returns the selected element(s) from the *range* slot.

Author(s)

The FLR Team

See Also

FLComp

Examples

```
# example FLStock
data(ple4)

range(ple4)

range(ple4, 'plusgroup')

range(ple4, 'plusgroup') <- 14
```

---

rgamma *Method rgamma*

Description

Random generation for the Gamma distribution with parameters 'shape' and 'scale'. 'shape' can be of class FLQuantPoint in which case 'shape' and 'scale' are set as $\hat{x}^2/\sigma^2$ and $\sigma^2/\hat{x}$.

Generic function

rgamma(n,shape,rate,scale)

Methods

signature(n=numeric,shape=FLQuantPoint,rate=missing,scale=missing) : Returns an FLQuant for Gamma-distributed values.

Author(s)

The FLR Team

See Also

rgamma, FLQuantPoint

Examples

```
flq <- FLQuant(rnorm(1000,mean=10,sd=2),dim=c(1,10,1,1,1,100))
flqp <- FLQuantPoint(flq)
rgamma(10,shape=flqp)
```

`rlnorm` *Method rlnorm*

Description

Random generation for the log normal distribution whose logarithm has mean equal to *meanlog* and standard deviation equal to *sdlog*. *meanlog* and *sdlog* can be given as FLQuant objects. If both are given as FLQuant objects their dimensions must be the same. If either of these arguments are FLQuant objects, rlnorm returns an FLQuant.

Generic function

rlnorm(n,meanlog,sdlog)

Methods

signature(n=numeric,meanlog=FLQuant,sdlog=FLQuant) : Generates random deviates for the log normal distribution. FLQuant arguments must have the same dimension. Returns an FLQuant object.

signature(n=numeric,meanlog=FLQuant,sdlog=numeric) : Generates random deviates for the log normal distribution. Returns an FLQuant object.

signature(n=numeric,meanlog=numeric,sdlog=FLQuant) : Generates random deviates for the log normal distribution. Returns an FLQuant object.

signature(n=numeric,meanlog=FLQuantPoint,sdlog=missing) : Generates random deviates for the log normal distribution. Returns an FLQuant object.

Author(s)

The FLR Team

See Also

rlnorm, FLQuant, FLQuantPoint

Examples

```
out <- rlnorm(1000, meanlog=FLQuant(c(5,5,5,5,5)),sdlog=FLQuant(c(0,1,2,3,4)))
apply(log(out),2,sd)
apply(log(out),2,mean)
```

rnorm *Method rnorm*

Description

Generates random numbers following a normal distribution. *mean* and *sd* can be specified as objects of class FLQuant, of the same dimensions, but any of the two could be given as a numeric. In this case the value will be reused accordingly.

Generic function

rnorm(n, mean, sd)

Methods

signature(n=numeric,mean=FLQuant,sd=FLQuant) : n is the number of iterations of the return object. mean and sd are FLQuant objects and must be of equal size. When both sd and mean are specified the returned object is an FLQuant object with n iterations, filled with randomly generated numbers. When only 1 of mean or sd is supplied the return object is a vector of length n.

signature(n=numeric,mean=numeric,sd=FLQuant) : same as above, but dimensions will be given by sd, and mean will be reused accordingly.

signature(n=numeric,mean=FLQuant,sd=numeric) : same as above, but instead dimensions will be given by mean, and sd will be reused accordingly.

signature(n=numeric,mean=FLQuantPoint,sd=missing) : uses an FLQuantPoint to obtain mean and sd from the mean and var iterations.

Author(s)

The FLR Team

See Also

rnorm, FLQuant, FLQuantPoint

Examples

```
data(ple4)
rnorm(10,mean=harvest(ple4)[,"2001"], sd=harvest(ple4)[,"2001"])
```

rpois *Method rpois*

Description

Generates random numbers following a Poisson distribution. *lambda*, the (non-negative) mean can be specified as an object of class FLQuant.

Generic function

rpois(n, lambda)

Methods

signature(n=numeric, lambda=FLQuant) : Generate a Poisson-distributed FLQuant object with 'n' iters.

Author(s)

The FLR Team

See Also

rpois, FLQuant

Examples

```
data(ple4)
rpois(50,lambda=harvest(ple4))
```

sd *Standard deviation of an FLModel object*

Description

sd computes the standard deviation of the parameter estimates in an FLModel object, either by calculating the diagonal of the square root of the variance-covariance matrix or, if multiple parameter estimates, as the standard deviation of each parameter.

Generic function

sd(x, na.rm)

Methods

signature(x=FLModel,na.rm=missing) : Standard deviation of a FLModel object

Author(s)

The FLR Team

See Also

sd, FLModel

Examples

```
data(nsher)
sd(nsher)
```

`setPlusGroup` *Method setPlusGroup*

Description

Calculates the appropriate values for the plusgroup of an object and returns a new object with the plusgroup set to the given age.

quant of the given object must be 'age', and the selected age must not be greater than the oldest age present in the object.

Generic function

sePlusGroup(x, plusgroup)

Methods

signature(x=FLQuant, plusgroup=numeric) : Adds values for the indicated age and older.

signature(x=FLStock, plusgroup=numeric) : The values for the plusgroup of the various slots in the `FLStock` object are calculated in different ways.

For slots catch.n, landings.n, discards.n and stock.n the plusgroup is calculated as the sum of values for ages equal to the plusgroup and above.

For slots catch.wt, landings.wt, discards.wt and stock.wt the plusgroup value is calculated as the weighted average of the values for ages equal to the plusgroup and above, weighted by the corresponding numbers at age.

If stock numbers at age are not available then the revised stock weights are calculated as a weighted average using the catch numbers at age and the slots for harvest, m, mat, harvest.spwn and m.spwn are truncated at the plusgroup age.

If stock numbers at age are available the plusgroup values for harvest, m, mat, harvest.spwn and m.spwn are calculated as a weighted average using the stock numbers at age.

signature(x=FLBiol, plusgroup=numeric) : Calculations are similar to FLStock above.

signature(x=FLCatch, plusgroup=numeric) : Abundances (*n* slot) are added, while weights (*wt*), natural mortality (*m*), and fecundity (*fec*) are averaged.

Author(s)

The FLR Team

See Also

FLStock, FLQuant, FLBiol, FLCatch

Examples

```
data(ple4)
ple4.pg <- setPlusGroup(ple4, 6)
```

---

show *Method show*

Description

Standard display of an object contents in an interactive session. Objects of class FLQuant with length > 1 along the sixth dimension (*iter*) are output in a summarised form, as median (mad), where mad is the median absolute deviation. See mad.

The same format is used for objects of class FLPar with length > 1 on the last dimension ('iter').

Generic function

show(object)

Methods

signature(object=FLQuant) : Describe method

signature(object=FLQuantPoint) : Describe method

signature(object=FLQuants) : Describe method

signature(object=FLPar) : Describe method

Author(s)

The FLR Team

See Also

FLComp

Examples

```
# no 'iter'
flq <- FLQuant(rnorm(80), dim=c(4,20), quant='age', units='kg')
flq

# with 'iter'
flq <- FLQuant(rnorm(800), dim=c(4,20,1,1,1,10), quant='age', units='kg')
flq
```

sop *Calculates the sum of products correction*

Description

Calculates the sum of products correction for quantities such a catch, discards, landings. For example in an object of class FLStock there are slots *catch.n*, *catch.wt* and *catch*. *catch* should equal the products of catch.n*catch.wt summed over ages. This function returns the ratio (i.e. the correction) of *catch.n*catch.wt* : *catch*, which can then be used to correct either *catch.n* or *catch.wt*.

Usage

```
sop(stock, slot)
```

Arguments

stock	An FLStock object
slot	Name of the slot group, i.e. "catch", "landings" or "discards" for an FLStock object.

Details

Can be used for any class or slot where there are the three FLQuant slots *foo*, *foo.n* and *foo.wt*, representing totals added over all quants (ages), numbers by quant, and weight by quant.

Value

Returns the ratio as an FLQuant

Author(s)

FLR Team

Examples

```
data(ple4)
sop(ple4,"catch")
```

splom *Method splom*

Description

Draws a conditional scatter plot matrix.

See the help page in lattice for a full description of each plot and all possible arguments.

Generic function

splom(x,data)

Methods

signature(x=FLPar,data=missing) : Conditional scatter plot matrix for all combinations of *params*.

Author(s)

The FLR Team

See Also

splom

Examples

```
data(nsher)
splom(params(nsher))
```

`spr0` *Method spr0*

Description

Calculates spawners per recruit at F=0.

This method currently does not work if any of the input objects have multiple units, seasons or areas (if dim(object)[3:5] > 1).

Generic function

quant(ssb, rec, fbar)

Methods

signature(ssb=FLQuant, rec=FLQuant, f=FLQuant) : `ssb`, `rec` and `fbar` as `FLQuant`(s)

signature(ssb=FLStock, rec=missing, f=missing) : `ssb`, `rec` and `fbar` are obtained from the slots of an `FLStock` object. `harvest` must have `units='f'`

signature(ssb=FLSR, rec=missing, f=FLQuant) : `rec` and `ssb` are obtained from an `FLSR` object, while `fbar` must be provided.

Author(s)

The FLR Team

See Also

FLStock, FLSR

Examples

```
data(ple4)
# example FLStock dataset
spr0(ple4)
```

sr *Stock-recruitment model function*

Description

The `sr()` function acts as a front end to the various functions available that implement fitting mechanisms for various stock/recruitment models. `fmle` is called if a likelihood function is present in the *logl* slot, otherwise `nls` is used instead.

Usage

```
sr(sr, ...)
```

Arguments

`sr`	An FLSR object.
`...`	Other parameters, depending on the model selected.

Value

An object of class `FLSR`

Author(s)

The FLR Team

See Also

`FLSR`

SRModelName *Convenience function to identify an SR model by its formula*

Description

A supplied formula, representing an stock-recruitment relationship, is matched against the list of all models defined in `FLCore` (See SRModels).

If a match is found, a tring character with the name of the model is returned, otherwise `FALSE` is obtained.

Usage

```
SRModelName(formula)
```

Arguments

`model`	A formula defining the model

Value

name	A character string or NULL

Author(s)

FLR Team

See Also

SRModels

Examples

```
SRModelName(rec ~ a * ssb * exp(-b * ssb))
```

SRModels *Methods SRModels Stock-Recruitment models*

Description

A range of stock-recruitment (SR) models commonly used in fisheries science are provided in FLCore.

Usage

```
ricker()
```

Arguments

rho	Autoregression
sigma2	Autoregression
obs	Observed values
hat	estimated values
steepness	Steepness.
vbiomass	Virgin biomass.
spr0	Spawners per recruit at F=0, see spr0.
model	character vector with model name, either 'bevholt' or 'ricker'.
model	A formula defining the model

Details

Each method is defined as a function returning a list with one or more elements as follows:

- model Formula for the model, using the slot names *rec* and *ssb*
- logl Function to calculate the loglikelihood of the given model when estimated through MLE (See fmle)
- initial Function to provide initial values for all parameters to the minimization algorithms called by fmle or nls. This function can also have two attributes, lower and upper, that give lower and upper limits for the parameter values, respectively. This is used by some of the methods defined in optim, like "L-BFGS-B"

. The *model*<- method for FLModel can then be called with *value* being a list as described above, the name of the function returning such a list, or the function itself. See the examples below.

Several functions to fit commonly-used SR models are available. They all use maximum likelihood to estimate the parameters through the method loglAR1.

- ricker: Ricker stock-recruitment model fit:

$$R = aSe^{-bS}$$

  a is related to productivity (recruits per stock unit at small stock size) and *b* to density dependence. (*a, b* > 0).

- bevholt: Beverton-Holt stock-recruitment model fit:

$$R = \frac{aS}{b+S}$$

  a is the maximum recruitment (asymptotically) and *b* is the stock level needed to produce the half of maximum recruitment $\frac{a}{2}$. (*a, b* > 0).

- segreg: Segmented regression stock-recruitment model fit:

$$R = \mathbf{ifelse}(S \leq b, aS, ab)$$

  a is the slope of the recruitment for stock levels below *b*, and ab is the mean recruitment for stock levels above *b*. (*a, b* > 0).

- geomean: Constant recruitment model fit, equal to the historical geometric mean recruitment.

$$exp(mean(log(R_1) + ... + log(R_n)))$$

- shepherd: Shepherd stock-recruitment model fit:

$$R = \frac{aS}{1 + (\frac{S}{b})^c}$$

  a represents density-independent survival (similar to *a* in the Ricker stock-recruit model), *b* the stock size above which density-dependent processes predominate over density-independent ones (also referred to as the threshold stock size), and *c* the degree of compensation.

- cushing: Cushing stock-recruitment model fit:

$$R = aSe^b$$

  This model has been used less often, and is limited by the fact that it is unbounded for *b*>=1 as *S* increases. (*a, b* > 0).

Stock recruitment models parameterized for steepness and virgin biomass:

- rickerSV: Fits a ricker stock-recruitment model parameterized for steepness and virgin biomass.

$$a = e^{\frac{b \cdot vbiomass}{spr0}}$$

$$b = \frac{\log(5 \cdot steepness)}{0.8 \cdot vbiomass}$$

- bevholtSV: Fits a Beverton-Holt stock-recruitment model parameterised for steepness and virgin biomass.

$$a = \frac{4 \cdot vbiomass \cdot steepness}{(spr0 \cdot (5 \cdot steepness - 1.0}$$

$$b = \frac{vbiomass(1.0 - steepness)}{5 \cdot steepnes - 1.0}$$

- sheperdSV: Fits a shepher stock-recruitment model parameterized for steepness and virgin biomass.

$$a = \frac{1.0 + (\frac{vbiomass}{b})^c}{spr0}$$

$$b = vbiomass(\frac{0.2 - steepness}{steepness(0.2)^c - 0.2})(\frac{-1.0}{c})$$

Models fitted using autoregressive residuals of first order:

- bevholtAR1, rickerAR1, segregAR1: Beverton-Holt, Ricker and segmented regression stock-recruitment models with autoregressive normal log residuals of first order. In the model fit, the corresponding stock-recruit model is combined with an autoregressive normal log likelihood of first order for the residuals. If R_t is the observed recruitment and $\hat{R}_t$ is the predicted recruitment, an autoregressive model of first order is fitted to the log-residuals, $x_t = \log(\frac{R_t}{\hat{R}_t})$.

$$x_t = \rho x_{t-1} + e$$

where e follows a normal distribution with mean 0: $e \sim N(0, \sigma^2_{AR})$.

Ricker model with one covariate. The covariate can be used, for example, to account for an enviromental factor that influences the recruitment dynamics. In the equations, c is the shape parameter and *X* is the covariate.

- rickerCa: Ricker stock-recruitment model with one multiplicative covariate.

$$R = a(1 - cX)Se^{-bS}$$

Value

`name` A character string or NULL

Generic function

quant(ssb, rec, fbar)

Author(s)

The FLR Team

The FLR Team

FLR Team

References

Beverton, R.J.H. and Holt, S.J. (1957) On the dynamics of exploited fish populations. MAFF Fish. Invest., Ser: II 19, 533.

Needle, C.L. Recruitment models: diagnosis and prognosis. Reviews in Fish Biology and Fisheries 11: 95-111, 2002.

Ricker, W.E. (1954) Stock and recruitment. J. Fish. Res. Bd Can. 11, 559-623.

Shepherd, J.G. (1982) A versatile new stock-recruitment relationship for fisheries and the construction of sustainable yield curves. J. Cons. Int. Explor. Mer 40, 67-75.

Seber, G.A.F., Wild, C.J. 2005. Autocorrelated Errors. In Seber, G.A.F., Wild, C.J. Nonlinear regression, pages 271-323. doi: 10.1002/0471725315.ch6.

See Also

FLSR, FLModel

FLStock, FLSR

SRModels

Examples

```
# inspect the output of one of the model functions
  bevholt()
  names(bevholt())
  bevholt()$logl

# once an FLSR model is in the workspace ...
  data(nsher)

# the three model-definition slots can be modified
# at once by calling 'model<-' with
# (1) a list
  model(nsher) <- bevholt()

# (2) the name of the function returning this list
  model(nsher) <- 'bevholt'
# or (3) the function itself that returns this list
  model(nsher) <- bevholt
```

---

ssb *Method ssb*

Description

Returns the Spawning Stock Biomass of FLStock and FLBiol objects.

For FLStock objects the nature of the calculation depends on the units in the harvest slot. See details below.

Generic function

ssb(object)

Methods

signature(object=FLStock) : If spawning occurs at the beginning of the year the calculated SSB is the same regardless of the units of the harvest slot. If spawning occurs at any other time during the year such that the stock is subject to fishing mortality prior to spawning then the calculated SSB will depend on the units of the harvest slot.

For an FLStock with harvest units 'f' SSB is calculated as

SSB = sum(N*exp(-F*propF-M*propM) * wt * mat)

For an FLStock with harvest units 'hr' SSB is calculated as

SSB = sum(N*(1-harvest)*exp(-M*propM) * wt * mat)

The units of the harvest slot in the FLStock object must be specified as either 'f' for an instantaneous fishing mortality or else as 'hr' for a harvest rate.

signature(object=FLBiol) : For an FLBiol the spawning biomass at the beginning of the year is calculated.

SSB = sum(N * wt * mat)

Author(s)

The FLR Team

See Also

FLBiol FLStock

Examples

```
data(ple4)
units(harvest(ple4)) # check the units of the harvest slot
ssb(ple4)
```

---

ssbpurec *Method ssbpurec*

Description

Calculates the Spawning Stock Biomass per unit recruit for an FLStock object.

The method calculates SSB per recruit at zero fishing mortality.

Generic function

ssbpurec(object)

Method arguments

`object :` an object of class FLStock.

`start, end :` The first and last year over which SSB per recruit is to be calculated. By default the first and last year of the FLStock object are used.

`type :` The type of calculation to perform. Currently only non-parm (non parametric) is supported whereby the SSB per recruit is calculated using mean values over the specified time period.

`recs :` The proportion of the year in which recruitment occurs.

`spwns :` The proportion of the year in which spawning occurs.

`plusgroup :` Is the last age a plusgroup TRUE/FALSE

Methods

signature(object=FLStock) : Returns the SBB-per-unit-recruit.

Author(s)

The FLR Team

See Also

FLStock

Examples

```
data(ple4)
ssbpurec(ple4)

ssbpurec(ple4, start=1980, end=2000)
```

---

ssn *Method ssn*

Description

Returns the Spawning Stock Numbers of FLBiol objects.

Generic function

ssn(object)

Methods

signature(object=FLBiol) : For a given FLBiol object, the spawning stock numbers are calculated as follows:

In Leslie matrix type models it is the spawning stock * numbers birthed (realised fecundity) that is the key recruitment driver, not SSB.

Author(s)

The FLR Team

See Also

FLComp FLStock

Examples

```
## Not run:
 data(ple4.biol)
 ssn(ple4.biol)

## End(Not run)
```

summary *Method summary*

Description

Outputs a general summary of the structure and content of the object. The particular output obtined depends on the class of the argument object.

Generic function

summary(object)

Methods

signature(object=FLQuant) : Returns dimensions, quant, units and distribution of data, including percentage of NAs

signature(object=FLQuantPoint) : Returns dimensions, quant, units and distribution of data

signature(object=FLComp) : Outputs name, desc, quant and dimensions of each slot

signature(object=FLQuants) : Returns dimensions, quant, units and distribution of data, including percentage of NAs for each element in the list

signature(object=FLPar) : Returns the values stored, or tehir basic statistics

signature(object=FLModel) : Returns the name, desc and content of model-related slots

Author(s)

The FLR Team

See Also

summary

Examples

```
data(ple4)
summary(ple4)

data(nsher)
summary(nsher)
```

survprob *Calculating survival probabilties given mortality in the FLBiol object*

Description

For an FLBiol object with the mortality-at-quant this method calculates the associated survival probability-at-quant. This can be later used by the r() method. The calculation can be carried out either by year or by cohort.

Usage

```
survprob(object, ...)
```

Arguments

object	An object of type FLBiol.
...	Extra arguments accepted by each implementation.

Details

Calculates the survival probability-at-quant given the mortality information in an FLBiol object - survival probability from one year to the next is simply exp(-M) and the survival probaiblity to a given quant is merely the product along the quant dimension of the individual survival probabilities.

Value

An object of class FLQuant.

Author(s)

FLR Team

See Also

FLBiol

Examples

```
## Not run:
data(nsher.biol)
nsh.ps <- survprob(nsh.biol,by='year')

## End(Not run)
```

sweep *Sweep out FLQuant Summaries*

Description

Return an FLQuant or FLCohort obtained from an input object by sweeping out a summary statistic along the selected dimensions.

Value

An FLQuant or FLCohort with the same shape as x, but with the summary statistics swept out.

Generic function

sweep(x, MARGIN, STATS, FUN="-", check.margin=TRUE, ...)

Method arguments

x an FLQuant or FLCohort object.

MARGIN a vector of indices giving the extents of x which correspond to STATS.

STATS the summary statistic which is to be swept out.

FUN the function to be used to carry out the sweep. In the case of binary operators such as "/" etc., the function name must backquoted or quoted. (FUN is found by a call to match.fun.)

check.margin logical. If TRUE (the default), warn if the length or dimensions of STATS do not match the specified dimensions of x. Set to FALSE for a small speed gain when you *know* that dimensions match.

... optional arguments to FUN.

Author(s)

The FLR Team

See Also

sweep

Examples

```
data(ple4)
mean.f <- apply(harvest(ple4),2,mean)
scaled.f <- sweep(harvest(ple4),2,mean.f,"/")
```

transform *Transform elements of a complex FLR object*

Description

Modification of individual elements of a complex FLR object can be carried out using `transform`. A series of named arguments, corresponding to the slots to modify can be provided to the method. Existing slots can be referred simply by its name on the right handside on the argument expressions (see example below).

Generic function

transform(_data, ...)

Methods

signature(_data=FLComp) : Method for all complex FLR classes that extend `FLComp`

Author(s)

The FLR Team

See Also

`transform`

Examples

```
data(ple4)
ple4 <- transform(ple4, m.spwn=m.spwn+0.2)
m.spwn(ple4)
```

trim *Trim FLR objects using named dimensions*

Description

Subsetting of FLR objects can be carried out using the dimension names by using `trim`. A number of dimension names and selected dimensions are passed to the method and those are used to subset the input object.

Exceptions are made for those classes where certain slots might differ in one or more dimensions. If trim is applied on an FLQuant object of length 1 in its first dimension and with dimension name equal to 'all', values to trim on specified for that dimension will be ignored. For example, `FLStock` objects contain slots with length=1 on their first dimension. Specifying values to trim along over the first dimension will have no effect on those slots (`catch`, `landings`, `discards`, and `stock`). Calculations might need to be carried out to recalculate those slots if their quant-structured counterparts are modified along the first dimension.

Generic function

trim(x)

Methods

signature(x=FLQuant) : Trims along the specified dimensions.

signature(x=FLComp) : Trims along the specified dimensions.

signature(x=FLStock) : Trims along the specified dimensions, but ignores the *quant* (first) dimension for those slots where it is always of length=1.

signature(x=FLCohort) : Trims along the specified dimensions.

signature(x=FLIndex) : Trims along the specified dimensions, but ignores the *quant* (first) dimension for thoe slots where it is of length=1.

Author(s)

The FLR Team

See Also

FLQuant, FLStock, FLCohort, FLIndex

Examples

```
data(ple4)

# This is equivalent to window(catch(ple4), start=1990, end=1995)
trim(catch(ple4), year=1990:1995)

trim(catch.n(ple4), year=1990:1995, age=1:2)

# Now on an FLStock
summary(trim(ple4), year=1990:1995)

# If 'age' is trimmed in ple4, catch, landings and discards need to be recalculated
shpl4 <- trim(ple4, age=1:4)
landings(shpl4) <- computeLandings(shpl4)
discards(shpl4) <- computeDiscards(shpl4)
catch(shpl4) <- computeCatch(shpl4)

summary(shpl4)
```

---

units *units attribute for FLQuant objects*

Description

Objects of class FLQuant contain an units attribute of class character. This should be used to store the corresponding units of measurement. This attribute can be directly accessed and modified using the units and units<- methods.

For complex objects, units will return a named list containing the attributes of all FLQuant slots. units of a complex object can be modified for all slots or a subset of them, by passing a named list with the new values. See examples below.

Generic function

units(x)

units<-(x,value)

Methods

signature(x=FLQuant) : Describe method

signature(x=FLComp) : Describe method

signature(x=FLPar) : Describe method

signature(x=FLCohort) : Describe method

Author(s)

The FLR Team

See Also

FLQuant, FLPar, FLCohort

Examples

```
flq <- FLQuant(rnorm(100), dim=c(5,20), units='kg')
units(flq)
units(flq) <- 't'
summary(flq)

# units for a complex object
data(ple4)
units(ple4)
units(ple4) <- list(harvest='hr')
```

---

uom *uom Units of Measurement*

Description

The 'units' attribute of FLQuant objects provides a mechanism for keeping track of the units of measurement of that particular piece of data.

Usage

```
uom(op, u1, u2)
```

Arguments

op	The arithmetic operator to be used, one of '+', '-', '*' or '/'
u1	The units of measurement string of the first object
u2	The units of measurement string of the second object

Details

Arithmetic operators for 'FLQuant' objects are aware of a limited set of units of measurement and will output the right unit when two object are arithmetically combined. For example, the product of object with units of 'kg' and '1000' will output an object with 'units' of 't' (for metric tonnes).

Operations involving combinations of units not defined will issue a warning, and the resulting 'units' attribute will simply keep a string indicating the input units of measurement and the operation carried out, as in '10 * 1000'.

Note that no scaling or modification of the values in the object takes place.

Conversion across units is carried out by the `uom()` function

Value

a string with the corresponding units of measurement, a string such as '10 *100' when not compatible

Recognized Units

The following units of measurement are recognized by the 'Arith' operators (+, -, * /).

Weight 'kg', 't'

Numbers 1 - 100000000, 1e0 - 1e8, 10^0 - 10^8

Mortality 'm', 'f', 'z', 'hr'

Other 'NA'

Author(s)

The FLR Team

See Also

`FLQuant` `units,FLArray-method`

Examples

```
# Conversion between weights
FLQuant(1, units='kg') * FLQuant(1000, units='1')

# Conversion between mortalities
FLQuant(0.2, units='m') + FLQuant(0.34, units='f')
```

---

update *Method update*

Description

`update` is a generic function for updating a model fitting using the same call that generated it. Input arguments can be provided that will alter the FLModel object accordingly.

Generic function

update(object, ...)

Methods

signature(object=FLModel) : Rerun using details(object)[['call']]

Author(s)

The FLR Team

See Also

FLModel, update

Examples

```
## Not run:
data(nsher)
nsher <- update(nsher, ssb=ssb(nsher) * 1.4)

## End(Not run)
```

---

var *Variance of an FLPar*

Description

`var` computes the variance of an `FLPar` object along the first dimension (`iter`) returning a value for each column (`param`)

By default, arguments `na.rm` and `use` have values of `FALSE` and `'all.obs'` respectively. See the `var` help page for more information on possible argument values.

Generic function

var(x, y, na.rm, use)

Methods

signature(x=FLPar,y=missing,na.rm=missing,use=missing) : Returns the variance of each parameter, computed along the *iter* dimension.

Author(s)

The FLR Team

See Also

`var`, `FLPar`

Examples

```
flp <- FLPar(rnorm(200), params=c('a', 'b'))
var(flp)
```

window *Extract time (year) windows of an FLR object*

Description

This method extracts a section of or extends an FLQuant or other FLR objects along the year dimension. If a frequency is specified, the new object contains data at only those year steps.

Although objects of class FLQuant do have another temporal dimension, season, currently window only works along the year dimension. To subset along other dimensions, refer to Extract-FLCore.

Generic function

window(x)

Methods

signature(x=FLQuant) : Subset along the year dimension

signature(x=FLComp) : The method is applied to each slot of class FLQuant.

signature(x=FLlst) : The window method is applied to each element in the list.

Author(s)

The FLR Team

See Also

window, Extract-FLCore

Examples

```
flq <- FLQuant(rnorm(50), dimnames=list(age=1:5, year=1990:2000))
window(flq, start=1995, end=1998)
window(flq, start=1990, end=2010, frequency=2)
```

wireframe *3D plot for FLQuant objects*

Description

Method to plot 3D representations of FLQuant objects

Usage

```
## S4 method for signature 'formula,FLQuant'
wireframe(x, data, ...)
```

Arguments

x	a formula formula for lattice
data	a FLQuant object with the values
...	Additional argument list to be passed to wireframe

Value

a wireframe plot

Examples

```
data(ple4)
wireframe(data~age+year, data=harvest(ple4))
```

Index