Package: FLCore (via r-universe)

January 20, 2025

Title Core Package of FLR, Fisheries Modelling in R

Version 2.6.20.9322

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/>.

X-schema.org-keywords fisheries, flr, R

License GPL (>= 2)

Depends R(>= 4.0), lattice, iterators

Imports graphics, grid, methods, Matrix, MASS, stats, stats4, utils, ggplot2

Suggests testthat, rlang, hedgehog

URL http://flr-project.org/FLCore

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

Collate 'genericMethods.R' 'uom.R' 'spread.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' 'FLStockR.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' 'SRmodels.R' 'coerce.R' 'PlotDiagnostics.R' 'jackknife.R' 'zzz.R' 'io.Adapt.R' 'io.VPA2Box.R' 'data.R' 'plot.R' 'oem.R' 'length.R'

LazyLoad Yes

LazyData No

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

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

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

RemoteRef HEAD

RemoteSha 9cfe884688fea5b04869ca16f344b3b606e3612a

Contents

`acc` *Catch curve estimates of total mortality at age (Z)*

Description

Catch curve estimates of total mortality at age (Z)

Usage

```
acc(object, ...)
```

Examples

```
data(ple4)
```

---

accessors *accessor and replacement methods for FLCore classes*

Description

All S4 classes defined in FLCore have methods for accessing and replacing any of their slots. These methods are named as the slot, and will return the content of the slot, for the accessor method, or modify it with the provided value.

Usage

```
name(object, ...)

desc(object, ...)

range(x, i) <- value

catch(object, ...)

catch.n(object, ...) <- value

catch.wt(object, ...)

discards(object, ...)

discards.n(object, ...)

discards.wt(object, ...)

landings(object, ...)

landings.n(object, ...)

landings.wt(object, ...)

m(object, ...)

stock(object, ...)

stock.n(object, ...)

stock.wt(object, ...)
```

```
m.spwn(object, ...)

harvest(object, catch, ...)

harvest.spwn(object, ...)

mat(object, ...)

n(object, ...)

m(object, ...)

wt(object, ...)

fec(object, ...)

spwn(object, ...)

effort(object, metier, ...)

type(object, ...)

distr(object, ...)

distribution(object, ...)

index(object, ...)

index.var(object, ...)

catch.n(object, ...)

catch.wt(object, ...)

sel.pattern(object, ...)

index.q(object, ...)

model(object, ...)

logl(object, ...)

gr(object, ...)

initial(object, ...)

logLik(object, ...)
```

```
vcov(object, ...) <- value

hessian(object, ...)

logerror(object, ...)

details(object, ...)

residuals(object, ...) <- value

fitted(object, ...)

rec(object, ...)

rec.obs(object, ...)

catch.q(object, ...)

discards.sel(object, ...)

landings.sel(object, ...)

params(object, ...)

## S4 replacement method for signature 'FLS,FLQuants'
catch(object) <- value
```

Arguments

object	The object from which a slot is to be extracted or replaced
value	Object to be inserted into the relevant slot

Details

Accessors and replacement methods, with some exception, are created at build time by calls to the createFLAccessors function. An accessor method is created for each slot, with simply calls slot() on the relevant slot name. For slots of class FLQuant, or FLArray-based, two methods are created: one if value is of class FLQuant, and another for value being a numeric vector. The later would insert the vector into the slot structure, using R's recycling rules.

Users are encouraged to use the accessor methods, rather than the '@' operator or the slot() method, to isolate code from the internal structure of the class. If a slot was to be altered or deleted in the future, a method would be provided to return the same value, computed from other slots.

Some of these methods might already not access directly an slot, and instead carry out a calculation to return the requested value, depending on the class being called with. Please refer to the particular method implementation to see if this is the case.

Accessor methods for slots of class predictModel behave differently depending on the compute argument. Please refer to the relevant help page for further clarification.

An object of class FLQuants, containing three elements named *catch*, *catch.n* and *catch.wt*, as returned by `computeCatch`, can be assigned directly to an object using *catch*<-.

Value

The required slot, for an accessor method, or invisible modifies the object, for the replacement one.

Author(s)

The FLR Team

See Also

`FLQuant`, `FLStock`, `FLIndex`, `FLBiol`, `predictModel`

Examples

```
data(ple4)

# To access the catch slot in an FLStock, use
catch(ple4)

# while to modify it, do
catch(ple4) <- catch(ple4) * 2

# A number can be used as input, to be recycled
m(ple4) <- 0.3
# same as a longer vector, by age
m(ple4) <- 0.4^(seq(1, 2, length=10))

# To see the methods defined by createFLAccessors, run, for example
getMethod('catch', 'FLS')

# Assign the 3 catch slots
catch(ple4) <- computeCatch(ple4, slot="all")
```

`adjust,FLStock-method` *Recalculate to adjust abundances to F and M*

Description

An FLStock object is projected forward using the initial abundances and the total mortality-at-age per timestep. New values for the stock.n and catch.n slots are calculated, assuming that harvest and m are correct. This calculation provides a test of the internal consistency of the object.

Usage

```
## S4 method for signature 'FLStock'
adjust(object)
```

Arguments

`object`	an `FLStock` object

Value

`FLStock` object

See Also

`harvest`

Examples

```
data(ple4)
test <- adjust(ple4)
# Difference in catch due to estimation error
plot(FLStocks(PLE=ple4, TEST=test))
```

`ageopt` *Age at which a cohort reaches its maximum biomass, calculated by year*

Description

The optimal (or critical) age is the transition point when a cohort achieves its maximum biomass in the absemce of fishing, i.e. losses due to natural mortality are now greater than gains due to increase in individual biomass.

Usage

```
## S4 method for signature 'FLStock'
ageopt(object)
```

Arguments

`object`	An object of class 'FLStock'

Value

The age at which maximum biomass is reached, an 'FLQuant'.

Author(s)

The FLR Team

See Also

FLStock

Examples

```
data(ple4)
ageopt(ple4)
```

AIC *Method AIC*

Description

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

Usage

```
## S4 method for signature 'FLModel,numeric'
AIC(object, k = 2)
```

Arguments

`object`	an FLModel object
`k`	the penalty per parameter to be used; the default 'k = 2' is the classical AIC.

Generic function

AIC(object, k)

Author(s)

The FLR Team

See Also

AIC, logLik, FLModel

Examples

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

append-FLCore *Append objects along the year dimension*

Description

Method to append objects along the *year* dimensions, by extending, combining and substituting sections of them.

Usage

```
## S4 method for signature 'FLQuant,FLQuant'
append(x, values, after = dims(values)$minyear - 1)

## S4 method for signature 'FLStock,FLStock'
append(x, values, after = dims(values)$minyear - 1)
```

Arguments

x	the object to which the values are to be appended to.
values	to be included in the modified object.
after	a year dimname after with the values are to be appended.

Details

FLR objects are commonly manipulated along the year dimension, and the append method offers a simple interface for substituting parts of an object with another, or combine them into one, extending them when necessary. The object to be included or added to the first will be placed as defined by the *year* dimnames, unless the *after* input argument specifies otherwise.

Attributes like dimnames and *units* will always be taken from the first argument, unless the necessary chnages to dimnames$year

Value

An object of the same class as x with values appended.

Author(s)

The FLR Team

See Also

base::append

Examples

```
# append(FLQuant, FLQuant)
fq1 <- FLQuant(1, dimnames=list(age=1:3, year=2000:2010))
fq2 <- FLQuant(2, dimnames=list(age=1:3, year=2011:2012))
fq3 <- FLQuant(2, dimnames=list(age=1:3, year=2014:2016))

# Appends by dimnames$year
append(fq1, fq2)
# Appends by dimnames$year with gap (2011:2013)
append(fq1, fq3)
# Appends inside x
append(fq1, fq2, after=2009)
# Appends after end of x
append(fq1, fq2, after=2013)

# append(FLStock, FLStock)
data(ple4)
fs1 <- window(ple4, end=2001)
fs2 <- window(ple4, start=2002)
fs3 <- window(ple4, start=2005)

# Appends by dimnames$year
stock.n(append(fs1, fs2))

# Appends by dimnames$year with gap (2011:2013)
stock.n(append(fs1, fs3))

# Appends inside x
stock.n(append(fs1, fs3, after=2000))
# Appends after end of x
stock.n(append(fs1, fs3, after=2005))
```

apply,FLArray,numeric,function-method

apply method for FLCore classes

Description

Applies a function over the margins of an array-based FLCore class

Usage

```
## S4 method for signature 'FLArray,numeric,function'
apply(X, MARGIN, FUN, ..., simplify = TRUE)

## S4 method for signature 'FLPar,ANY,ANY'
apply(X, MARGIN, FUN, ..., simplify = TRUE)

## S4 method for signature 'FLQuantJK,numeric,function'
```

```
apply(X, MARGIN, FUN, ..., simplify = TRUE)

## S4 method for signature 'FLParJK,numeric,function'
apply(X, MARGIN, FUN, ..., simplify = TRUE)
```

Details

These methods call R's base::apply on an FLArray the standard arithmetic operators included in the `Arith` group ("+", "-", "*", ‘"^", "%%", "%/%", and "/"), so that they return an object of the appropriate class.

When the operation involves objects of two classes (e.g. `FLPar` and `FLQuant`), the class is the returned object is that of the more complexs object, in this case `FLQuant`.

Author(s)

The FLR Team

See Also

base::apply

Examples

```
flq <- FLQuant(rlnorm(90), dim=c(3,10), units='kg')
flp <- FLPar(a=99)

# FLQuant and numeric
flq * 25
# Two FLQuant objects
flq + flq
```

---

ar1rlnorm *Generates a time series of possible bias-corrected lognormal autocorrelated random values*

Description

Thorston, 2020.

Usage

```
ar1rlnorm(
  rho,
  years,
  iters = 1,
  meanlog = 0,
  sdlog = 1,
  bias.correct = TRUE,
  ...
)
```

Arguments

rho	Autocorrelation coefficient.
years	Vector of year names.
iters	Number of iterations.
meanlog	Mean of the series in log space.
sdlog	Marginal standard deviation in log space.
bias.correct	Should bias-correction be applied? Defaults to TRUE.

Value

An FLQuant object

Author(s)

Iago Mosqueira (WMR), Henning Winker (JRC).

References

Thorson, J. T. Predicting recruitment density dependence and intrinsic growth rate for all fishes worldwide using a data-integrated life-history model. Fish Fish. 2020; 21: 237– 251. https://doi-org.ezproxy.library.wur.nl/10.1111/faf.12427

See Also

rlnorm

Examples

```
devs <- ar1rlnorm(rho=0.6, years=2000:2030, iter=500, meanlog=0, sdlog=1)
plot(devs)
```

Arith,numeric,FLArray-method

Arithmetic operators for FLCore classes

Description

Overloaded arithmetic operators for FLCore classes

Usage

```
## S4 method for signature 'numeric,FLArray'
Arith(e1, e2)

## S4 method for signature 'FLArray,numeric'
Arith(e1, e2)

## S4 method for signature 'FLArray,FLArray'
Arith(e1, e2)

## S4 method for signature 'FLPar,FLPar'
Arith(e1, e2)

## S4 method for signature 'FLArray,FLPar'
Arith(e1, e2)

## S4 method for signature 'FLPar,FLArray'
Arith(e1, e2)
```

Details

These methods apply the standard arithmetic operators included in the `Arith` group ("+", "-", "*", "^", "%%", "%/%", and "/"), so that they return an object of the appropriate class.

When the operation involves objects of two classes (e.g. `FLPar` and `FLQuant`), the class is the returned object is that of the more complexs object, in this case `FLQuant`.

Author(s)

The FLR Team

See Also

methods::Arith base::Arithmetic

Examples

```
flq <- FLQuant(rlnorm(90), dim=c(3,10), units='kg')
flp <- FLPar(a=99)

# FLQuant and numeric
flq * 25
# Two FLQuant objects
flq + flq

# FLQuant and FLPar
flq / flp
```

as.FLSRs — *Convert an FLStock into a list of one or FLSR objects.*

Description

A single FLStock can be coerced into a list with one or more objects of class FLSR, each of them typically set to a diefferemt stock-recruit model.

Usage

```
as.FLSRs(x, models = NULL, ...)
```

Arguments

x	An estimated FLStock object to coerce.
models	Name(s) of model(s) to fit.
...	Any extra arguments to be passed to *as.FLSR*.

Value

An objecdt of class FLSRs

Author(s)

FLR Team, 2023.

See Also

FLSRs FLSRs as.FLSR()

Examples

```
data(ple4)
as.FLSRs(ple4, model=c("bevholt", "segreg"))
```

bias — *Bias of estimates through jackknife*

Description

Description: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque eleifend odio ac rutrum luctus. Aenean placerat porttitor commodo. Pellentesque eget porta libero. Pellentesque molestie mi sed orci feugiat, non mollis enim tristique.

Usage

```
## S4 method for signature 'FLQuantJK'
bias(x)

## S4 method for signature 'FLParJK'
bias(x)
```

Arguments

`x` An object holding estimates obtained through jackknife

Details

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

$$\widehat{Bias}_{(\theta)} = (n-1)((\frac{1}{n}\sum_{i=1}^{n}\hat\theta_{(i)}) - \hat\theta)$$

Value

A value for the mean bias

Author(s)

The FLR Team

See Also

FLComp

Examples

```
flq <- FLQuant(1:8)
flj <- jackknife(flq)
bias(flj)
```

BIC *Method 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.

Usage

```
## S4 method for signature 'FLModel'
BIC(object)
```

Arguments

`object`	a fitted FLModel object for which there exists a 'logLik' method to extract the corresponding log-likelihood.

Generic function

BIC(object)

Author(s)

The FLR Team

See Also

AIC, BIC, FLModel, logLik

Examples

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

bubbles *Method Bubbles plot*

Description

This method plots three dimensional data such as matrices by age and year or age-class, very common in fisheries. The area of each bubble is proportional to the corresponding value in the matrix. Note that `bubbles` accepts an argument `bub.scale` to control the relative size of the bubbles. Positive and negative values have separate colours.

Usage

```
## S4 method for signature 'formula,FLQuant'
bubbles(x, data, bub.scale = 2.5, col = c("blue", "red"), ...)

## S4 method for signature 'formula,data.frame'
bubbles(x, data, bub.scale = 2.5, col = c("blue", "red"), ...)

## S4 method for signature 'formula,FLCohort'
bubbles(x, data, bub.scale = 2.5, ...)

## S4 method for signature 'formula,FLQuants'
bubbles(x, data, bub.scale = 2.5, bub.col = gray(c(0.1, 0.1)), ...)
```

Generic function

bubbles(x, data)

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"), pch=16)
```

catch.n,FLQuant-method

catch.n calculation method

Description

Calculate catch.n (catch-at-age/length) from abundances, F and M using the catch equation

Usage

```
## S4 method for signature 'FLQuant'
catch.n(object, harvest, m)
```

Details

The catch-at-age/length, commonly found in the catch.n slot of an `FLStock` object, can be simply calculated from abundances-at-age/length, and natural and fishing mortalities-at-age/length by applying the catch equation

$$C = N \cdot F \frac{F}{M+F} \cdot (1 - \mathrm{e}^{(} - M - F))$$

Author(s)

The FLR Team

See Also

FLStock

Examples

```
data(ple4)
res <- catch.n(stock.n(ple4), harvest(ple4), m(ple4))
catch.n(ple4) / res
```

`catchInmature` *Proportion of mature and inmature fish in the catch*

Description

The proportion in weight of mature and inmature fish in the catch can be computed using catchMature and catchInmature.

Usage

```
catchInmature(object)

catchMature(object)
```

Arguments

`object`	An age-structured FLStock object

Value

An FLQuant object

Author(s)

The FLR Team

See Also

FLComp

Examples

```
data(ple4)
catchInmature(ple4)
catchMature(ple4)
```

`coerce-methods` *Convert Objects Between Classes*

Description

Objects of various **FLCore** classes can be converted into other classes, both basic R ones, like `data.frame`, and others defined in the package. For the specifics of the precise calculations carried out for each pair of classes, see below.

Arguments

`object`	Object to be converted.
`Class`	Name of the class to convert the object to, `character`.

Value

An object of the requested class.

FLArray to data.frame

The six dimensions of an `FLArray` are converted into seven columns, named `quant` (or any other name given to the first dimension in the object), `year`, `unit`, `season`, `area`, `iter` and `data`. The last one contains the actual numbers stored in the array. `units` are stored as an attribute to the `data.frame`. The `year` and `data` columns are of type `numeric`, while all others are `factor`.

FLPar to data.frame

The two or more dimensions of an *FLPar* objects are converted into three or more columns. For a 2D objects, they are named *params*, *iter* and *data*. The last one contains the actual numbers stored in the array, in a column type `numeric`, while all others are `factor`.

Author(s)

The FLR Team

See Also

base::as, base::coerce

Examples

```
# from FLQuant to data.frame
as(FLQuant(rnorm(100), dim=c(5, 20)), "data.frame")
# from FLPar to data.frame
as(FLPar(phi=rnorm(10), rho=rlnorm(10)), "data.frame")
```

`compare` *A method for comparing FLR objects*

Description

Comparisons of complete objects of FLR classes can be carried out and a report table is generated to better identify differences. Comparisons do not substitute but complement those provided by R's all.equal and identical.

Usage

```
compare(result, target, ...)
```

Arguments

`result`	First element in comparison, result of method or operation.
`target`	Second element, desired output.

Value

A table of comparisons, of class data.frame.

Author(s)

Iago Mosqueira (WMR)

`compute` *Methods to compute quantities*

Description

Methods to compute total quant-aggregated catch, landings, discards and stock biomass from age or length-structured numbers and mean weights.

Methods to compute total quant-aggregated catch, landings, discards and stock biomass from age or length-structured numbers and mean weights.

Usage

```
computeLandings(object, ...)

computeDiscards(object, ...)

computeCatch(object, ...)

computeStock(object, ...)

computeHarvest(object, catch, ...)

computeLandings(object, ...)

computeDiscards(object, ...)

computeCatch(object, ...)

computeStock(object, ...)

## S4 method for signature 'FLS'
computeLandings(object, na.rm = TRUE)

## S4 method for signature 'FLS'
```

```
computeDiscards(object, na.rm = TRUE)

## S4 method for signature 'FLS'
computeCatch(object, slot = "catch", na.rm = TRUE)

## S4 method for signature 'FLS'
computeStock(object, na.rm = TRUE)
```

Details

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 landings, discards 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 (from landings and discards) 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 element names `catch`, `catch.n` and `catch.wt`, which can then be passed directly to the `catch<-` replacement method.

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 landings, discards 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 (from landings and discards) 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 element 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, ...)

computeCatch(object, ...)

computeLandings(object, ...)

computeDiscards(object, ...)

computeStock(object, ...)

Author(s)

The FLR Team

See Also

FLComp

FLComp

Examples

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

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

computeHarvest,FLStock,missing-method

Computes fishing mortality from abundances, catches and natural mortality

Description

Objects or class ’FLStock’ already contain a ’harvest’ slot to store estimates of fishing mortality at age, for example those obtained from a stock assessment method. Fishing mortality at age can be recalculated using two methods:

Usage

```
## S4 method for signature 'FLStock,missing'
computeHarvest(object, units = NULL)
```

Arguments

`units`	Harvest to be computed as ’f’ or ’hr’, ’character’.
`x`	An object of class ’FLStock’.

Value

An 'FLQuant' with the calculated fishing mortalities at age.

Author(s)

The FLR Team

See Also

FLStock harvest() FLQuant

Examples

```
data(ple4)
# Compute 'f' from stock.n and Baranov
computeHarvest(ple4)
# Recomputes all F at age by solving catch Baranov
recomputeHarvest(ple4)
```

cpue *cpue, a method to generate an observation of a CPUE index of abundance*

Description

The observation of stock abundance by CPUE series from commercial fleets is an important step in the generation of management advice that needs to replicated on an Operating Model during any simulation exercise. This method gemnerates an observation of biomass or numbers-at-age from an FLstock being used as OM.

Usage

```
cpue(object, index, ...)

## S4 method for signature 'FLStock,missing'
cpue(
  object,
  sel.pattern = harvest(object),
  effort = units(harvest(object)),
  biomass = TRUE
)
```

Arguments

object	The object from which to generate the observation.
effort	Units of index to use to mimic effort series in the fishery, "f" or "hr"
sel	The selectivity of the survey, defaults to be 1 for all ages.
mass	Is the index to be in weight at age?

Value

An FLQuant for the index of abundance, age-disaggregated

Author(s)

Laurie Kell & Iago Mosqueira, FLR Team.

See Also

FLComp

Examples

```
data(ple4)

cpue(ple4)
# Am aggregated biomass CPUE
quantSums(cpue(ple4))

## Not run:
plot(FLQuants(om=stock(ple4), cpue=quantSums(cpue(ple4)),
  hr=quantSums(cpue(ple4, effort="hr"))))

## End(Not run)
```

---

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

datasets *FLCore datasets*

Description

Example datasets for the classes defined in FLCore.

Details

- ple4, FLStockA dataset for North Sea (ICES Area IV) plaice. Catch, landings, discards, natural mortality, weight-at-age and maturity, together with the VPA estimated abundances and fishing mortalities.
- ple4sex, FLStockA 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.
- ple4.index, FLIndexA dataset of North Sea (ICES Area IV) plaice survey catch per unit effort, index and index variance.
- ple4.indices, FLIndicesA dataset of three North Sea (ICES Area IV) plaice survey catch per unit effort series. Index and index variance.
- ple4.biol, FLBiolA dataset of the North Sea plaice population. Numbers, natural mortality, mass and fecundity-at-age.
- nsher , FLSRStock and recruit data and fitted relationship for autumn spawning 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

Examples

```
data(ple4)
summary(ple4)

data(nsher)
is(nsher)
```

dbind — *Methods for binding objects of array classes along a given dimension*

Description

These methods can bind two or more objects of array-based classes (e.g. FLQuant), along the specified dimension.

Usage

```
dbind(x, y, ...)

## S4 method for signature 'FLArray,FLArray'
dbind(x, y, ..., dim = 1)

qbind(...)

ybind(...)

ubind(...)

sbind(...)

abind(...)

ibind(...)
```

Arguments

x	First object to bind
y	Second object to bind
...	Other objects to bind
dim	Dimension to bind on, *numeric* or *character*.

Details

The objects to bind must contain the same dimmames in all dimensions other than that used to bind, while dimnames in the selected one must differ. See the examples below for correct and incorrect uses.

Object are bound in the order they are provided, with no attempt to sort according to the dimnames of the chosen dimension.

The implementation is based around a single method (*dbind*), that operates along the dimension position or name indicated by the *dim* argument. A series of shortcut functions call the method for specific dimensions, with names related to the dimensions name they operate on (e.g. ybind for *year*).

Value

An object of the same class as the inputs

Author(s)

Iago Mosqueira (EC JRC)

See Also

FLQuant FLArray

Examples

```
# By iter
x <- FLQuant(rnorm(80000), dim=c(4,20,1,1,1,1000))
y <- FLQuant(rnorm(80000), dim=c(4,20,1,1,1,1000))
  dimnames(y) <- list(iter=1001:2000)
ibind(x,y)

# By quant (age)
x <- FLQuant(1, dimnames=list(age=1:3, year=1:10))
y <- FLQuant(2, dimnames=list(age=4:12, year=1:10))
qbind(x, y)

# By year
x <- FLQuant(1, dimnames=list(age=1:3, year=1:10))
y <- FLQuant(2, dimnames=list(age=1:3, year=11:20))
z <- FLQuant(3, dimnames=list(age=1:3, year=21:30))
ybind(x, y, z)

# By season
x <- FLQuant(1, dimnames=list(year=1:10, season=1:2))
y <- FLQuant(2, dimnames=list(year=1:10, season=3:4))
sbind(x, y)
```

dims *Method dims*

Description

List with information on object dimensions

List with information on object dimensions

Usage

```
dims(obj, ...)

dims(obj, ...)
```

```
## S4 method for signature 'FLQuant'
dims(obj, element, ...)
```

Details

Method `dims` returns a named list with information on the dimensions and dimension names of a given object. 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 dimension

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 Length of the third (`unit`) dimension

season Length of the fourth (`season`) dimension

area Length of the fifth (`area`) dimension

iter Length 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 in the input object. Use `quant` to obtain the name and extract the relevant element from the result list.

Method `dims` returns a named list with information on the dimensions and dimension names of a given object. 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 dimension

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 Length of the third (`unit`) dimension

season Length of the fourth (season) dimension

area Length of the fifth (area) dimension

iter Length 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 in the input object. Use quant to obtain the name and extract the relevant element from the result list.

Generic function

dims(obj)

dims(obj)

Author(s)

The FLR Team

See Also

dimnames, FLQuant

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)]]
```

---

dimSummaries *Summaries by dimension*

Description

Methods to compute various summary calculations (sum, mean, variance) over selected dimensions of objects from any array-based classes (e.g. FLQuant). These methods return an object of the same dimensions as the input but with length one in the dimension chosen to operate along.

Usage

```
quantSums(x, ...)

yearSums(x, ...)

unitSums(x, ...)

seasonSums(x, ...)

areaSums(x, ...)

iterSums(x, ...)

dimSums(x, ...)

quantMeans(x, ...)

yearMedians(x, ...)

yearMeans(x, ...)

unitMeans(x, ...)

seasonMeans(x, ...)

areaMeans(x, ...)

iterMeans(x, ...)

dimMeans(x, ...)

quantVars(x, ...)

yearVars(x, ...)

unitVars(x, ...)

seasonVars(x, ...)

areaVars(x, ...)

iterVars(x, ...)

dimVars(x, ...)

iterMedians(x, ...)

iterCVs(x, ...)
```

```
iterProb(x, ...)

## S4 method for signature 'FLQuant'
quantSums(x, na.rm = TRUE)

## S4 method for signature 'FLQuant'
yearSums(x, na.rm = TRUE)

## S4 method for signature 'FLQuant'
unitSums(x, na.rm = TRUE)

## S4 method for signature 'FLQuant'
seasonSums(x, na.rm = TRUE)

## S4 method for signature 'FLQuant'
areaSums(x, na.rm = TRUE)

## S4 method for signature 'FLQuant'
iterSums(x, na.rm = TRUE)

## S4 method for signature 'FLQuant'
quantMeans(x, na.rm = TRUE)

## S4 method for signature 'FLQuant'
yearMeans(x, na.rm = TRUE)

## S4 method for signature 'FLQuant'
unitMeans(x, na.rm = TRUE)

## S4 method for signature 'FLQuant'
seasonMeans(x, na.rm = TRUE)

## S4 method for signature 'FLQuant'
areaMeans(x, na.rm = TRUE)

## S4 method for signature 'FLQuant'
iterMeans(x, na.rm = TRUE)

## S4 method for signature 'FLQuant'
yearMedians(x, na.rm = TRUE)

## S4 method for signature 'FLQuant'
iterMedians(x, na.rm = TRUE)

## S4 method for signature 'FLQuant'
quantVars(x, na.rm = TRUE)
```

```
## S4 method for signature 'FLQuant'
yearVars(x, na.rm = TRUE)

## S4 method for signature 'FLQuant'
unitVars(x, na.rm = TRUE)

## S4 method for signature 'FLQuant'
seasonVars(x, na.rm = TRUE)

## S4 method for signature 'FLQuant'
areaVars(x, na.rm = TRUE)

## S4 method for signature 'FLQuant'
iterVars(x, na.rm = TRUE)

## S4 method for signature 'FLQuant'
iterCVs(x, na.rm = TRUE)

## S4 method for signature 'FLQuant'
iterProb(x, na.rm = TRUE)

## S4 method for signature 'FLQuantDistr'
yearSums(x, na.rm = TRUE)

## S4 method for signature 'FLQuantDistr'
unitSums(x, na.rm = TRUE)

## S4 method for signature 'FLQuantDistr'
seasonSums(x, na.rm = TRUE)

## S4 method for signature 'FLQuantDistr'
areaSums(x, na.rm = TRUE)

## S4 method for signature 'FLQuantDistr'
yearMeans(x, na.rm = TRUE)

## S4 method for signature 'FLQuantDistr'
unitMeans(x, na.rm = TRUE)

## S4 method for signature 'FLQuantDistr'
seasonMeans(x, na.rm = TRUE)

## S4 method for signature 'FLQuantDistr'
areaMeans(x, na.rm = TRUE)

## S4 method for signature 'FLQuantDistr'
iterMeans(x, na.rm = TRUE)
```

```
## S4 method for signature 'FLQuantDistr'
iterMedians(x, na.rm = TRUE)

## S4 method for signature 'FLQuantDistr'
quantVars(x, na.rm = TRUE)

## S4 method for signature 'FLQuantDistr'
yearVars(x, na.rm = TRUE)

## S4 method for signature 'FLQuantDistr'
unitVars(x, na.rm = TRUE)

## S4 method for signature 'FLQuantDistr'
seasonVars(x, na.rm = TRUE)

## S4 method for signature 'FLQuantDistr'
areaVars(x, na.rm = TRUE)

## S4 method for signature 'FLQuantDistr'
iterVars(x, na.rm = TRUE)

## S4 method for signature 'FLPar'
iterMeans(x, na.rm = TRUE)

## S4 method for signature 'FLPar'
iterMedians(x, na.rm = TRUE)

## S4 method for signature 'FLPar'
iterVars(x, na.rm = TRUE)

## S4 method for signature 'FLPar'
iterSums(x, na.rm = TRUE)
```

Arguments

`x`	An object.
`na.rm`	Should NAs be removed before calculation? Defaults to TRUE.

Details

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). Medians and CVs can also be computed along the sixth dimension, `iter`.

These methods encapsulate a call to `apply` with the corresponding dimensions and function: `mean`, `median`, `var`, and `sum`, while `iterCVs` are computed as `sqrt(iterVars) / iterMeans`.

In contrast with R standard behaviour, the sum of a dimension where all elements are `NA` will be `NA` and not 0. See example below.

Methods working along the iter dimension are also defined for objects of class `FLPar`.

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

Generic methods

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), iterMedians(x), iterSums(x) dimSums(x), dimMeans(x), dimVars(x)

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))

# NA dims stay as NA when summed along
x <- FLQuant(c(NA, NA, NA, rnorm(6)), dim=c(3, 3))
quantSums(x)
# although in fact a sum of no elements (as na.rm=TRUE) is zero
apply(x, 2:6, sum, na.rm=TRUE)
```

`discardsRatio` *Compute the ratio of discards to total catch in numbers or weight*

Description

A calculation is made of the proportion of discards over total catch at age, either as numbers (value = ’numbers’) or weight (value = ’weight’), or for the total discards and catch in biomass (value = ’total’).

Usage

```
discardsRatio(object, value = c("numbers", "weight", "total"))
```

Arguments

object	An object of class 'FLStock'
value	One of 'numbers' (default), 'weight' or 'total'.

Value

The discards ratio (between 0 and 1), 'FLQuant'

Author(s)

The FLR Team

See Also

FLStock

Examples

```
data(ple4)
# Discards ratio at age in numbers
discardsRatio(ple4)
# Total proportion of discards by year
discardsRatio(ple4, value="total")
```

drop,FLArray-method *drop method for FLCore array-based classes*

Description

Delete the dimensions of an array which have only one level.

Usage

```
## S4 method for signature 'FLArray'
drop(x)
```

Details

This method calls R's base::drop on the `@.Data` slot of an FLArray. Dimensions of length one are thus dropped, as is the class attribute and the `units` slot, and an array of equal or less dimensions, a matrix or a vector is returned.

On an FLQuant object with

Author(s)

The FLR Team

See Also

base::drop

Examples

```
x <- FLQuant(1:3, dim=c(3,3))
drop(x)
is(drop(x))
dim(drop(x))

# Result of drop can be used for matrix algebra
# for example to calculate aging error

data(ple4)
aging.error <- diag(0.8, 10)
diag(aging.error[-1,]) <- c(rep(0.1, 8), 0.2)
diag(aging.error[, -1]) <- c(0.2, rep(0.1, 8))
t(aging.error) %*% drop(catch.n(ple4))
```

---

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 is part of.

Usage

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

Arguments

`object`	The FLR S4 over whicvh the predictModel evaluation should take place
`slot`	The predictModel object to be evaluated

Value

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

Author(s)

The FLR Team

See Also

predictModel

`exp,FLQuant-method` *exp and log methods FLCore array-based classes*

Description

Compute the exponential and logarithmic functions

Usage

```
## S4 method for signature 'FLQuant'
exp(x)

## S4 method for signature 'FLQuant'
log(x, ...)
```

Details

This method simply calls R's base::exp and base::drop, but take care of returning the right units of measurement, that is "" or character(1).

Author(s)

The FLR Team

See Also

base::exp base::log

Examples

```
x <- FLQuant(c(4,2,7,4,2,9), units="1000")
log(x)
units(log(x))
```

`Extract` *Extract*

Description

Extract or replace parts of an FLR Object

Usage

```
## S4 method for signature 'FLArray,ANY,ANY,ANY'
x[i, j, k, l, m, n, ..., drop = FALSE]

## S4 method for signature 'FLArray,array,missing,missing'
x[i]

## S4 replacement method for signature 'FLArray,ANY,ANY,ANY'
x[i, j, k, l, m, n, ...] <- value

## S4 replacement method for signature 'FLArray,ANY,ANY,FLArray'
x[i, j, k, l, m, n, ...] <- value

## S4 method for signature 'FLQuant'
x$name

## S4 method for signature 'FLQuantDistr,ANY,ANY,ANY'
x[i, j, k, l, m, n, drop = FALSE]

## S4 method for signature 'FLQuantDistr,array,missing,missing'
x[i]

## S4 method for signature 'FLPar,ANY,ANY,ANY'
x[i, j, k, l, m, n, ..., drop = FALSE]

## S4 method for signature 'FLPar,array,missing,missing'
x[i]

## S4 replacement method for signature 'FLPar,ANY,ANY,ANY'
x[i, j, k, l, m, n, ...] <- value

## S4 method for signature 'FLPar'
x$name

## S4 replacement method for signature 'FLPar'
x$name <- value

## S4 method for signature 'FLComp,ANY,ANY,ANY'
x[i, j, k, l, m, n, ..., drop = FALSE]

## S4 replacement method for signature 'FLComp,ANY,ANY,ANY'
x[i, j, k, l, m, n, ...] <- value

## S4 method for signature 'FLStock,ANY,ANY,ANY'
x[i, j, k, l, m, n, ..., drop = FALSE]

## S4 replacement method for signature 'FLStock,ANY,ANY,FLStock'
x[i, j, k, l, m, n, ...] <- value
```

```
## S4 method for signature 'FLI,ANY,ANY,ANY'
x[i, j, k, l, m, n, ..., drop = FALSE]

## S4 method for signature 'predictModel,ANY,missing,ANY'
x[i, k, l, m, n, ..., drop = FALSE]

## S4 replacement method for signature 'FLlst,ANY,missing'
x[[i, j]] <- value

## S4 replacement method for signature 'FLlst'
x$name <- value

## S4 replacement method for signature 'FLlst,ANY,missing,ANY'
x[i, j] <- value

## S4 method for signature 'FLlst,ANY,missing,ANY'
x[i, drop]
```

Arguments

x	object from which to extract or replace element(s)
i, j, k, l, m, n	indices specifying elements to extract or replace on any of the six dimensions.
...	indices specifying elements to extract or replace by dimension name.
drop	If 'TRUE' the result is coerced to the lowest possible dimension, and so might change class (e.g. drop='TRUE' on an FLQuant might return an array of less dimensions, a matrix or a vector.
value	An object of the same class, or simpler if drop=TRUE, than 'x'.
name	See Extract for further details.

Details

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

Please note the differences between referencing sections of an object by position using values of class numeric, or by using 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.

'$' for the FLPar and FLQuant classes operate only along the first dimension ('params' or 'quant'), and are provided to be used specially in formulas.

Generic function

x,i,j,drop

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

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

\$<-(x,name,value)

Author(s)

The FLR Team

See Also

Extract

Examples

```
flq <- FLQuant(rnorm(200), dimnames=list(age=0:4, year=1991:2000,
  season=1:4))

# Extracting by position...
  flq[1,]
  flq[,1:5]
  flq[1:2,,,c(1,3)]

# ...by dimnames
  flq['0',]
  flq[,'1991']
  flq[,as.character(1991:1995),,'1']

# Dimensions of length one can be drop
  flq[1, drop=TRUE]

# Replacing part of the object
  flq['0',,,1]<-0
```

ffwd *Project forward an FLStock for a fbar target*

Description

Projection of an FLStock object for a fishing mortality target does not always require the features of fwd().Fast-forward an FLStock object for a fishing mortality yearly target only.

Usage

```
ffwd(object, sr, fbar = control, control = fbar, deviances = "missing")
```

Arguments

`object`	An *FLStock*
`sr`	A stock-recruit relationship, *FLSR* or *predictModel*.
`fbar`	Yearly target for average fishing mortality, *FLQuant*.
`control`	Yearly target for average fishing mortality, *fwdControl*.
`deviances`	Deviances for the strock-recruit relationsip, *FLQuant*.

Value

The projected *FLStock* object.

Author(s)

Iago MOSQUEIRA (MWR), Henning WINKEL (JRC).

See Also

fwd

Examples

```
data(ple4)
sr <- predictModel(model=bevholt, params=FLPar(a=140.4e4, b=1.448e5))
# Project for fixed Fbar=0.21
run <- ffwd(ple4, sr=sr, fbar=FLQuant(0.21, dimnames=list(year=1958:2017)))
plot(run)
```

`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.

Usage

```
FLBiol(object, ...)

## S4 method for signature 'FLQuant'
FLBiol(object, plusgroup = dims(object)$max, ...)
```

Arguments

object	FLQuant object used for sizing
...	Other objects to be assigned by name to the class slots
plusgroup	Plusgroup age, to be stored in range

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.

Accessors

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

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

Constructor

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

Validity

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

Iters The dimname for iter1 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.

Usage

```
FLBiols(object, ...)

## S4 method for signature 'FLBiol'
FLBiols(object, ...)

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

## S4 method for signature 'list'
FLBiols(object, ...)
```

Arguments

`object`	unnamed object to be added to the list
`...`	other named or unnamed 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

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

`FLlst`, `list`, `vector`

`FLCohort` *Class FLCohort*

Description

A class for modelling cohorts.

Usage

```
FLCohort(object, ...)

## S4 method for signature 'FLQuant'
FLCohort(object, ...)

## S4 method for signature 'FLCohort'
FLCohort(object, units = units(object))

## S4 method for signature 'array'
FLCohort(
  object,
  dim = rep(1, 6),
  dimnames = "missing",
  units = "NA",
  iter = 1,
  fill.iter = TRUE
)

## S4 method for signature 'vector'
FLCohort(
  object,
  dim = c(length(object), rep(1, 5)),
  dimnames = "missing",
  units = "NA",
  iter = 1
)

## S4 method for signature 'missing'
FLCohort(object, dim = rep(1, 6), dimnames = "missing", units = "NA", iter = 1)
```

Arguments

`object`	Input numeric object
`...`	Additonal arguments

Details

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`

Constructor

Objects of this class are generally constructed from an FLQuant object.

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 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.

Usage

```
FLCohorts(object, ...)
```

Arguments

object	unnamed object to be added to the list
...	other named or unnamed objects

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

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

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 iter1 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. Elements are specific to each child class.

Author(s)

The FLR Team

See Also

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

FLComps *Class FLComps*

Description

A virtual class that forms the basis for many FLR list classes. No objects of this class can be constructed.

Arguments

`object`	unnamed object to be added to the list
`...`	other named or unnamed objects

Validity

Elements All elements must be of a class that inherits from FLComp

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`.

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

FLlst, FLComp

FLI Class FLI

Description

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

Arguments

`object`	FLQuant object used for sizing
`...`	Other objects to be assigned by name to the class slots

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`)

Accessors

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

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

Constructor

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

Validity

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

Iters The dimname for iter1 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 for modelling abundance indices.

Usage

```
FLIndex(object, ...)

## S4 method for signature 'FLQuant'
FLIndex(object, plusgroup = dims(object)$max, ...)

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

Details

The `FLIndex` object 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 Named numeric vector containing the quant and year ranges, the plusgroup, and the period of the year, expressed as proportions of a year, that corresponds to the index (`numeric`).

Author(s)

The FLR Team

See Also

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

Examples

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

# Creat an FLIndex object using an existing FLQuant object.
  data(ple4)
  # Create a perfect index of abundance from abundance at age
    fli2 <- FLIndex(index=stock.n(ple4))
  # Add some noise around the signal
    index(fli2) <- index(fli2)*exp(rnorm(1, index(fli2)-index(fli2), 0.1))
```

FLIndexBiomass *Class FLIndexBiomass*

Description

A class for modelling biomass indices.

Usage

```
FLIndexBiomass(object, ...)

## S4 method for signature 'FLQuant'
FLIndexBiomass(object, plusgroup = dims(object)$max, ...)

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

Arguments

`object`	FLQuant object used for sizing
`...`	Other objects to be assigned by name to the class slots

Details

The `FLIndexBiomass` object holds data and parameters related to biomass 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`)

Accessors

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

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

Constructor

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

Validity

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

Iters The dimname for iter1 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 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.

Usage

```
FLIndices(object, ...)

## S4 method for signature 'FLI'
FLIndices(object, ...)

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

## S4 method for signature 'list'
FLIndices(object, ...)
```

Arguments

object	unnamed object to be added to the list
...	other named or unnamed objects

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.

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

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 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.

Usage

```
FLlst(object, ...)
```

Arguments

`object`	unnamed object to be added to the list
`...`	other named or unnamed objects

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`.

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

[, [<-, [[<-, $<-, 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*

Description

A virtual class for statistical models

Usage

```
FLModel(model, ...)
```

Details

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 are stored in slots created by a specified class based on FLModel. See for example FLSR for a class used for stock-recruitment models.

The initial slot contains a function used to obtain initial values for the numerical solver. It can also contain two attributes, upper and lower that limit the sarch area for each parameter.

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.

distribution Associated error probability dfistribution, factor.

fitted Estimated values, 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`.

Author(s)

The FLR Team

See Also

AIC, BIC, fmle, nls, FLComp

Examples

```
# Normally, FLModel objects won't be created if "class" is not set
  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 class FLGrowth with a single new slot, 'mass'
    setClass('FLGrowth', representation('FLModel', mass='FLArray'))

  # Define a creator method based on FLModel
   setGeneric("FLGrowth", function(object, ...) standardGeneric("FLGrowth"))
    setMethod('FLGrowth', signature(object='ANY'),
      function(object, ...) return(FLModel(object, ..., class='FLGrowth')))
    setMethod('FLGrowth', signature(object='missing'),
      function(...) return(FLModel(formula(NULL), ..., class='FLGrowth')))

  # Define an accessor method
    setMethod('mass', signature(object='FLGrowth'),
      function(object) return(slot(object, 'mass')))
```

FLModelSim *Class FLModelSim*

Description

A virtual class for statistical simulation models

Usage

```
FLModelSim(object, ...)

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

Details

The `FLModelSim` 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 are stored in slots created by a specified class that is based on `FLModelSim`. 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 `FLModelSim`, including `fmle`, `nls-FLCore` and `glm`.

Slots

params Estimated parameter values. `FLPar`.

distr `character`

vcov `array`

model `formula`

Author(s)

The FLR Team

See Also

AIC, BIC, fmle, nls

FLModelSims *Class FLModelSims*

Description

A list of FLModelSim objects.

Usage

```
FLModelSims(object, ...)

## S4 method for signature 'ANY'
FLModelSims(object, ...)

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

## S4 method for signature 'list'
FLModelSims(object)

## S4 method for signature 'FLModelSims'
FLModelSims(object)
```

Arguments

object	unnamed object to be added to the list
...	other named or unnamed objects

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.

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

FLlst, list, vector

FLPar *Class FLPar*

Description

A class for storing parameters of a model.

Usage

```
FLPar(object, ...)
```

Details

The `FLPar` class 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 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`.

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'))

FLPar(rnorm(20), dimnames=list(params=c('a','b'), year=1990:1999, iter=1),
  units='NA')

# with iters
  FLPar(rnorm(80), params=c('a', 'b'), iter=1:40)
```

FLParJK *Class FLParJK*

Description

A class for storing parameters of a jackknifed model fit.

Usage

```
## S4 method for signature 'ANY'
FLParJK(object, orig)

## S4 method for signature 'FLParJK'
orig(object)
```

Slots

`.Data` Jackknifed object, `FLPar`.

`units` units of measurement, `character`.

`orig` original object being jackknifed, `FLPar`.

Validity

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

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

Objects of this class are commonly created by calling the `jackknife()` method A construction method exists for this class that can take named arguments for any of its slots. All slots are then created to match the requirements of the class validity.

Author(s)

The FLR Team

See Also

`FLPar`

FLPars *Class FLPars*

Description

A list of FLPar objects.

Usage

```
FLPars(object, ...)

## S4 method for signature 'ANY'
FLPars(object, ...)

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

## S4 method for signature 'list'
FLPars(object)

## S4 method for signature 'FLPars'
FLPars(object)
```

Arguments

object	unnamed object to be added to the list
...	other named or unnamed 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

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

FLlst, list, vector

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.

Usage

```
FLQuant(object, ...)

## S4 method for signature 'missing'
FLQuant(
  object,
  dim = rep(1, 6),
  dimnames = "missing",
  quant = NULL,
  units = "NA",
  iter = 1
)

## S4 method for signature 'vector'
FLQuant(
  object,
  dim = rep(1, 6),
  dimnames = "missing",
  quant = NULL,
  units = "NA",
  iter = 1,
  fill.iter = TRUE
)

## S4 method for signature 'array'
FLQuant(
  object,
  dim = rep(1, 6),
  dimnames = "missing",
  quant = NULL,
  units = "NA",
  iter = 1,
  fill.iter = TRUE
)

## S4 method for signature 'matrix'
FLQuant(object, dim = lapply(dimnames, length), dimnames = "missing", ...)
```

```
## S4 method for signature 'FLQuant'
FLQuant(
  object,
  quant = attributes(object)[["quant"]],
  units = attributes(object)[["units"]],
  dimnames = attributes(object)[["dimnames"]],
  iter = dim(object)[6],
  fill.iter = TRUE,
  dim = attributes(object)[["dim"]]
)
```

Arguments

`object`	Input numeric object
`...`	Additional arguments
`dim`	Vector of dimension lengths
`dimnames`	List of dimension names
`quant`	Character vector for name of first dimension
`units`	Character vctor of units of measurement, see uom
`iter`	Number of iterations, i.e. length of the 6th dimension
`fill.iter`	Should iterations be filled with the same content as the first?

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)


FLQuant()
summary(FLQuant())

FLQuant(1:10)

FLQuant(array(rnorm(9), dim=c(3,3,3)))
FLQuant(matrix(rnorm(12), nrow=4, ncol=3))

FLQuant(FLQuant(array(rnorm(9), dim=c(3,3,3)), units='kg'), units='t')
```

FLQuantDistr *A class for samples of a probability distribution*

Description

This extended FLQuant class holds both a measure of central tendendy (mean, median) and of dispersion (tipically variance), to be later used to generate, for example, random numbers with those mean and variances.

Usage

```
FLQuantDistr(object, var, ...)

## S4 method for signature 'ANY,ANY'
FLQuantDistr(object, var, ...)

## S4 method for signature 'FLQuant,FLQuant'
FLQuantDistr(object, var, units = object@units, distr = "norm")
```

Arguments

`object`	Input numeric object
`...`	Additonal arguments

Slots

.Data Unnamed slot for storing the mean (or other measure of expectation) (`FLQuant`).

var Variance, or other measure of dispersion, (`FLQuant`).

distr Name of the probability distribution, see Details (`character`).

Validity

slot dims .Data and var slots must have the same dimensions.

slot dimnames .Data and var slots must have the same dimnames.

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

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.

The contents of the unnamed slot (.Data) can be accessed through the `e()` method, see Example below.

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 the .Data slot.

Methods

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

Arith .

Arithmetic

The methods under the *Arith* group have been defined for objects of this class, both for operations between two `FLQuantDistr` objects and with objects of class `FLQuant` (`FLArray`) as follows:

+, FLQuantDistr,FLArray .

-, FLQuantDistr,FLArray .

, *FLQuantDistr,FLArray*. \item/, FLQuantDistr,FLArray. \item+, FLQuantDistr,FLQuantDistr. \item-, FLQuantDistr
.

Author(s)

The FLR Team

See Also

FLQuant

Examples

```
data(ple4)
fqd <- FLQuantDistr(catch.n(ple4), var=catch.n(ple4) * 10, distr='norm')
```

`FLQuantJK` *A class for jackknifing fisheries data*

Description

This extended FLQuant class holds both a jackknifed FLQuant, one in which each iter is missing one element, and the original object, as a separate `FLQuant` in the `orig` slot.

Usage

```
## S4 method for signature 'ANY'
FLQuantJK(object, orig)

## S4 method for signature 'FLQuantJK'
orig(object)

## S4 method for signature 'FLQuants'
orig(object)
```

Arguments

`object`	Input numeric object
`...`	Additonal arguments

Slots

.Data Unnamed slot containing the jackknifed object(FLQuant).

orig Original object, (FLQuant).

Validity

slot dims .Data and orig slots must have the same dimensions 1-5.

slot dimnames .Data and var slots must have the same dimnames 1-5.

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

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

Objects of this class must be constructed from an FLQuant that is to be jackknifed, through the `jackknife` method.

Methods

All methods defined for the FLQuant class are available, but they will operate only on the jackknifed (`.Data`) slot. Please use `orig()` to apply them to the original object stored in the class.

Author(s)

The FLR Team

See Also

FLQuant

Examples

```
data(ple4)
fjk <- jackknife(stock(ple4))
# New object has as many iters as length of jackknifed dimension (defaults to 'year')
dim(fjk)
```

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.

Usage

```
FLQuantPoint(object, ...)

## S4 method for signature 'missing'
FLQuantPoint(..., units = "NA", n = 1)

## S4 method for signature 'FLQuant'
FLQuantPoint(object, ..., probs = c(0.25, 0.75))

## S4 method for signature 'FLQuantPoint'
n(object, ...)
```

Arguments

`object`	Input numeric object
`...`	Additonal arguments

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).

quantile: returns the 'lowq' or 'uppq' iter, depending on the value of 'probs' (0.25 or 0.75).

Constructor

Inputs can be of class:

`FLQuant`**:** An FLQuant object with iters (i.e. dim6 > 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(rlnorm(2000), dim=c(10,20,1,1,1,200), units="kg")
flqp <- FLQuantPoint(flq)
flqp <- FLQuantPoint(flq, probs=c(0.05, 0.95))
summary(flqp)
mean(flqp)
var(flqp)
rnorm(200, flqp)
```

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.

Usage

```
FLQuants(object, ...)

## S4 method for signature 'ANY'
FLQuants(object, ...)

## S4 method for signature 'FLComp'
FLQuants(object, ...)

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

## S4 method for signature 'list'
FLQuants(object, ...)

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

Arguments

object	unnamed object to be added to the list
...	other named or unnamed objects

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`.

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

Examples

```
# Compute various FLStock indicators
  data(ple4)
  fqs <- FLQuants(ssb=ssb(ple4), catch=catch(ple4), rec=rec(ple4),
    f=fbar(ple4))
  summary(fqs)
  xyplot(data~year|qname, fqs, type='b', scales=list(relation='free'))
```

`FLS` *Class FLS*

Description

A virtual class that forms the basis for the `FLStock` and `FLStockLen` classes. No objects of this class can be constructed.

Validity

None No particular validity checks

Slots

catch Total catch weight (`FLQuant`).

catch.n Catch numbers (`FLQuant`).

catch.wt Mean catch weights (`FLQuant`).

desc Description of the stock (`character`).

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`).

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

mat Proportion mature (`FLQuant`).

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

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

name Name of the stock (`character`).

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

Author(s)

The FLR Team

See Also

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

FLSR *Class FLSR*

Description

Class for stock-recruitment models.

Usage

```
FLSR(model, ...)

## S4 method for signature 'ANY'
FLSR(model, ...)

## S4 method for signature 'missing'
FLSR(model, ...)
```

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)
```

FLSRs *FLSRS 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 FLSRs must all be of class FLSR. It implements a lock mechanism that, when turned on, does not allow the user to increase or decrease the object length.*

Description

FLSRS 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 FLSRs must all be of class FLSR. It implements a lock mechanism that, when turned on, does not allow the user to increase or decrease the object length.

Usage

```
FLSRs(object, ...)

## S4 method for signature 'FLSR'
FLSRs(object, ...)

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

## S4 method for signature 'list'
FLSRs(object, ...)
```

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.

Author(s)

The FLR Team

See Also

FLlst, list, FLSR

Examples

```
data(nsher)
bnsher <- nsher
model(bnsher) <- bevholt
```

```
bnsher <- fmle(bnsher)
fls <- FLSRs(Ricker=nsher, BevHolt=bnsher)
summary(fls)
```

FLStock *Class FLStock*

Description

A class for modelling a fish stock.

Usage

```
FLStock(object, ...)

## S4 method for signature 'FLQuant'
FLStock(object, plusgroup = dims(object)$max, ...)

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

## S4 method for signature 'FLQuants'
FLStock(object, ...)
```

Arguments

`object`	FLQuant object used for sizing
`...`	Other objects to be assigned by name to the class slots
`plusgroup`	Plusgroup age, to be stored in range

Details

The `FLStock` object contains a representation of a fish stock as constructed for the purposes of scientific analysis and advice. 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 this slot should be set to 'hr' 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 the stock (`character`).

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

Accessors

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

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

Constructor

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

Author(s)

The FLR Team

See Also

[, [<-, as.FLBiol, 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)
summary(ple4)

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

# perform similar calculation as the preceding apply function
  discards(ple4) <- computeDiscards(ple4)
  catch(ple4) <- computeCatch(ple4)
  catch(ple4) <- computeCatch(ple4, slot="all")

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

# subset and trim the FLStock
  ple4[,1]
  trim(ple4, age=2:6, year=1980:1990)

# Calculate SSB, and SSB per recruit at zero fishing mortality
  ssb(ple4)
  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.

Usage

```
FLStockLen(object, ...)

## S4 method for signature 'FLQuant'
FLStockLen(object, ...)

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

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 this slot 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 the stock (`character`).

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

Author(s)

The FLR Team

See Also

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

Examples

```
stkl <- FLStockLen(m=FLQuant(0.2, dimnames=list(len=seq(5, 50, by=2), year=2015:2020)))
summary(stkl)
# Unnamed FLQuant used for sizing
stkl <- FLStockLen(FLQuant(0.2, dimnames=list(len=seq(5, 50, by=2), year=2015:2020)))
summary(stkl)
m(stkl)
```

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.

Usage

```
FLStocks(object, ...)

## S4 method for signature 'FLStock'
FLStocks(object, ...)

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

## S4 method for signature 'list'
FLStocks(object, ...)
```

Arguments

object	unnamed object to be added to the list
...	other named or unnamed objects

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.

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

plot, FLlst, list

Examples

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

FUNCTION *Extract and modify the recruitment time series*

Description

Recruitment in number of fish is the first row of the 'stock.n' slot of an age-structured 'FLStock'. These convenience functions allow a clearer syntax when retrieving of altering the content of 'stock.nrec.age,', where 'rec.age' is usually the first age in the object.

Usage

```
## S4 method for signature 'FLStock'
rec(object, rec.age = as.character(object@range["min"]))
```

Arguments

`object`	An object of class 'FLStock'
`rec.age`	What age to extract, defaults to first one. As 'character' to select by name or as 'numeric' by position.

Value

RETURN Lorem ipsum dolor sit amet

Author(s)

The FLR Team

See Also

FLComp

Examples

```
data(ple4)
rec(ple4)
# Multiple recruitment by a factor of 2
rec(ple4) <- rec(ple4) * 2
```

`Funwanted` *Calculate the discards and landings-associated fishing mortalities*

Description

Computes the fishing mortality at age (harvest) associated with either landings (*Fwanted*) or discards (*Funwanted*) through the respective proportions at age. The function names reflect the convention used in ICES.

Usage

```
Funwanted(x, ages = dimnames(x)$age)
```

Arguments

`x`	An FLStock object, with harvest
`ages`	Ages over which the respective Fbar calculation applies

Value

An FLQuant

Examples

```
data(ple4)
Fwanted(ple4, ages=2:6)
Funwanted(ple4, ages=1:3)
```

`fwdWindow` *Extend a FLR object along the year dimension and set future assumed values*

Description

Objects to be projected into the future are extended until an end year, and the values of certain quantities, usually assume constant, are set following different mechanisms.

Usage

```
fwdWindow(x, y, ...)

## S4 method for signature 'FLStock,missing'
fwdWindow(
  x,
  end = dims(x)$maxyear,
  nsq = 3,
  fun = c("mean", "sample"),
 years = list(wt = nsq, mat = nsq, m = nsq, spwn = nsq, discards.ratio = nsq, catch.sel
    = nsq)
)
```

Arguments

x	The FLR object to extend.
y	A second object from which information is taken.

Details

For ’FLStock’

Value

An object of the same class as ’x’.

Author(s)

The FLR Team.

See Also

window()

Examples

```
data(ple4)
# Use mean of last three years and extend until 2020
fut <- fwdWindow(ple4, end=2020)
# Check values on catch.wt
catch.wt(fut)[, ac(2015:2020)]
# Use mean of the 2010:2015 period
fut <- fwdWindow(ple4, end=2020, years=2010:2015)
# Use last three years mean, but last five for 'wt'
fut <- fwdWindow(ple4, end=2020, nsq=3, years=list(wt=5))
stock.wt(fut)[, ac(2013:2020)]
catch.sel(fut)[, ac(2013:2020)]
# Resample from last years for 'wt'
fut <- fwdWindow(ple4, end=2020, nsq=3, fun=c(wt='sample'))
# Years to resample can be different for 'catch.sel'
```

```
fut <- fwdWindow(ple4, end=2020, nsq=3,
  fun=c(wt='sample', catch.sel='sample'), years=c(wt=10, catch.sel=5))
# 'wt' slot has been resampled,
stock.wt(fut)[, ac(2015:2020)]
# while others have used a 3 year average
catch.sel(fut)[, ac(2015:2020)]
```

---

getSlotNamesClass *Names of slots of a given class*

Description

This function returns the names, as a character vector, of the slots in an S4 object that are of the class specified by the 'class' argument. Comparison is done using is(), so class inheritance is matched.

Usage

```
getSlotNamesClass(object, class)
```

Arguments

`object`	An S4 object to check slots from.
`class`	The name of the class to match, 'character'.

Author(s)

The FLR Team

Examples

```
data(ple4)
getSlotNamesClass(ple4, 'FLQuant')
```

---

group *Group objects over some index by applying a function over a single dimension*

Description

Array objects (e.g. FLQuant or FLQuants) are divided along a single dimnension following a given index or expression, an aggregating function is applied to each subset, and the results are joined again. Data can be added, for example, by decade or for two age groups.

Usage

```
group(x, FUN, ...)

## S4 method for signature 'FLQuant,function'
group(x, FUN = sum, ...)
```

Arguments

x	An object to group.
FUN	A function to apply along the chosen dimension, defaults to 'sum'.
...	An expression or indexing vector, named as the chosen dimension. Extra arguments to FUN can also be provided, but cannmot match names in x.

Value

A single object with reduced dimensionality.

Author(s)

Iago Mosqueira (WMR)

Examples

```
data(ple4)
# Add catch-at-age along two age groups, 'juv'eniles and 'adu'lts
group(catch.n(ple4), sum, age=c('juv', 'juv', rep('adu', 8)))
# An expression can use based on dimnames
group(catch.n(ple4), sum, age=age < 3)
# Mean by lustrum, by using 'year - year %% 5'
group(catch.n(ple4), mean, year = year - year %% 5)
```

iav *Compute the inter-annual variability of a time series*

Description

The inter-annual variability of a time series stored in an FLQuant object, is computed as $|x_y - x_{y-1})/x_{y-1}|$. The resulting object will be one year shorter than the input. The first year will be missing as values are assigned to the final year of each pair.

Usage

```
iav(object)
```

Value

An object of the same class as object.

Author(s)

The FLR Team

Examples

```
data(ple4)
# Compute inter-annual variability in catch
iav(catch(ple4))
```

---

indicators.len *Calculate quantile(s) of length distribution*

Description

z = (k * (linf - lmean)) / (lmean - lc) lmean = sum(naa * len) / sum(naa) lc, length at first capture

Usage

```
indicators.len(
  object,
  indicators = "lbar",
  model = vonbert,
  params,
  cv = 0.1,
  lmax = 1.25,
  bin = 1,
  n = 500,
  metric = catch.n,
  ...
)

lenquantile(x, quantile = 0.5)

lmax5(x)

l95(x)

l25(x)

lc50(x)

lmode(x)

lbar(x)

lmean(x)
```

```
lmaxy(x, lenwt)

pmega(x, linf, lopt = linf * 2/3)

bheqz(x, linf, k, t0, lc = lc50(x))
```

References

- Kell, L.T., Minto, C., Gerritsen, H.D. 2022. Evaluation of the skill of length-based indicators to identify stock status and trends. ICES Journal of Marine Science. doiu: 10.1093/icesjms/fsac043.
- ICES. 2015. Report of the Fifth Workshop on the Development of Quantitative Assessment Methodologies based on Life-history Traits, Exploitation Characteristics and other Relevant Parameters for Data-limited Stocks (WKLIFE V), 5–9 October 2015, Lisbon, Portugal. ICES CM 2015/ACOM:56. 157 pp.
- ICES. 2020. Tenth Workshop on the Development of Quantitative Assessment Methodologies based on LIFE-history traits, exploitation characteristics, and other relevant parameters for data-limited stocks (WKLIFE X). ICES Scientific Reports. 2:98. 72 pp. http://doi.org/10.17895/ices.pub.5985

Examples

```
data(ple4)
indicators.len(ple4, indicators=c('lbar', 'lmaxy'),
  params=FLPar(linf=132, k=0.080, t0=-0.35), metric='catch.n',
  lenwt=FLPar(a=0.01030, b=2.975))
indicators.len(ple4, indicators=c('pmega'),
  params=FLPar(linf=60, k=2.29e-01, t0=-1.37), metric='catch.n')
data(ple4.index)
indicators.len(ple4.index, indicators=c('lbar', 'lmean'),
  params=FLPar(linf=132, k=0.080, t0=-0.35), metric='index')
#
ialk <- invALK(params=FLPar(linf = 60, k = 2.29e-01, t0 = -1.37e+00),
  model=vonbert, age=1:10, lmax=1.2)
samps <- lenSamples(catch.n(ple4), invALK=ialk, n=250)
lenquantile(samps, 0.50)
lmax5(samps)
l95(samps)
l25(samps)
lc50(samps)
lmode(samps)
lbar(samps)
lmean(samps)
# Linf(ple4) = 60
lmean(samps) / (0.75 * lc50(samps) + 0.25 * 60) #
lenwt <- FLPar(a=0.01030, b=2.975)
lmaxy(samps, lenwt)
pmega(samps, linf=60)
linf <- 60
k <- 2.29e-01
t0 <- -1.37e+00
bheqz(samps, linf = 60, k = 2.29e-01, t0 = -1.37e+00)
```

`intersect` *Returns FLR objects trimmed to their shared dimensions.*

Description

Objects sharing certain dimensions, as inferred by their *dimnames*, are subset to the common ones along all dimensions. The returned object is of one of the *FLlst* classes, as corresponds to the input class. The objects in the list can then be, for example, combined or directly compared, as shown in the examples.

Usage

```
## S4 method for signature 'FLArray,FLArray'
intersect(x, y)
```

Arguments

`x`	First object to be compared and subset
`y`	Second object to be compared and subset

Value

And object of the corresponding *FLsdt*-based plural class.

Author(s)

The FLR Team

See Also

base::intercept

Examples

```
big <- FLQuant(64.39, dimnames=list(age=1:4, year=2001:2012))
small <- FLQuant(3.52, dimnames=list(age=2:3, year=2001:2005))
intersect(big, small)

# Two FLQuant objects can be added along their common dimension using Reduce()
Reduce('+', intersect(big, small))
```

iter *Methods iter*

Description

Select or modify iterations of an FLR object

Usage

```
iter(obj, ...)

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

Details

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

In complex objects 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 iteration 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)

Author(s)

The FLR Team

See Also

FLComp, FLQuant

Examples

```
# For an FLQuant
  flq <- FLQuant(rnorm(800), dim=c(4,10,2), iter=10)
  iter(flq, 2)
# For the more complex FLStock object
  data(ple4)
  fls <- propagate(ple4, 10)
  # Extraction using iter...
    fls2 <- iter(fls, 2)
    summary(fls2)
```

jackknife *Method jackknife*

Description

Jackknife resampling

Usage

```
## S4 method for signature 'FLQuant'
jackknife(object, dim = "year", na.rm = TRUE)

## S4 method for signature 'FLQuants'
jackknife(object, ...)

## S4 method for signature 'FLModel'
jackknife(object, slot)
```

Details

The `jackknife` method sets up objects ready for jackknifing, i.e. to systematically recompute a given statistic leaving out one observation at a time. From this new set of "observations" for the statistic, estimates for the bias and variance of the statstic can be calculated.

Input objects cannot have length > 1 along the `iter` dimension, and the main slot in the resulting object will have as many `iters` as the number of elements in the original object that are not `NA`.

Generic function

jackknife(object, ...)

Author(s)

The FLR Team

See Also

`FLQuantJK` `FLParJK`

Examples

```
flq <- FLQuant(1:8)
flj <- jackknife(flq)
iters(flj)
```

join *Joins objects along a dimensions where dimnames differ*

Description

FLQuant objects are joined along a single dimension, on which dimnames are different. This is the reverse operation to divide.

Usage

```
join(x, y, ...)

## S4 method for signature 'FLQuant,FLQuant'
join(x, y)

## S4 method for signature 'FLQuants,missing'
join(x, y)
```

Arguments

x	An object to join
y	An object to join

Value

A single object

Author(s)

Iago Mosqueira (WMR)

Examples

```
data(ple4)
# JOIN over age dimension
x <- catch.n(ple4)[1,]
y <- catch.n(ple4)[2,]
join(x, y)
# JOIN over year dimension
x <- catch.n(ple4)[,10:20]
y <- catch.n(ple4)[,21:25]
join(x, y)
div <- divide(catch.n(ple4), dim=1)
is(div)
length(div)
join(div)
all.equal(join(divide(catch.n(ple4), dim=1)), catch.n(ple4))
```

lattice *Lattice methods*

Description

Implementation of Trellis graphics in FLR

Usage

```
## S4 method for signature 'formula,FLQuant'
xyplot(x, data, ...)

## S4 method for signature 'formula,FLCohort'
xyplot(x, data, ...)

## S4 method for signature 'formula,FLQuants'
xyplot(x, data, ...)

## S4 method for signature 'formula,FLComp'
xyplot(x, data, ...)

## S4 method for signature 'formula,FLQuant'
bwplot(x, data, ...)

## S4 method for signature 'formula,FLComp'
bwplot(x, data, ...)

## S4 method for signature 'formula,FLQuant'
dotplot(x, data, ...)

## S4 method for signature 'formula,FLComp'
dotplot(x, data, ...)

## S4 method for signature 'formula,FLQuant'
barchart(x, data, ...)

## S4 method for signature 'formula,FLComp'
barchart(x, data, ...)

## S4 method for signature 'formula,FLQuant'
stripplot(x, data, ...)

## S4 method for signature 'formula,FLComp'
stripplot(x, data, ...)

## S4 method for signature 'formula,FLQuant'
histogram(x, data, ...)
```

```
## S4 method for signature 'formula,FLComp'
histogram(x, data, ...)

## S4 method for signature 'formula,FLQuants'
histogram(x, data, ...)

## S4 method for signature 'formula,FLPar'
densityplot(x, data, ...)
```

Details

Plot methods in the `lattice` package are available for an object of classes `FLQuant`, `FLQuants` or those derived from `FLComp`.

See the help page in `lattice` for a full description of each plot method and 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, ...)

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'))
```

mase *Compute mean absolute scaled error (MASE)*

Description

Franses, PH. "A note on the Mean Absolute Scaled Error". International Journal of Forecasting. 32 (1): 20–22. doi:10.1016/j.ijforecast.2015.03.008.

Usage

```
mase(ref, preds, ...)

## S4 method for signature 'FLQuant,FLQuants'
mase(ref, preds, order = c("inverse", "ahead"))

## S4 method for signature 'FLIndices,list'
mase(ref, preds, order = "inverse", wt = "missing")
```

Arguments

ref	Reference or naive prediction.
preds	Predicitions to compare to reference.
...	Extra arguments.
order	Are predictions in 'inverse' (default) or 'ahead' order.
wt	Mean weights-at-age to use with indices.

Value

A numeric vector of the same length as 'preds'.

meanage — *Calculate the mean age in the stock and catch*

Description

Average age in the stock numbers or catch-at-age.

Usage

```
meanage(object)

meanageCatch(object)
```

Arguments

object	An age-structured FLStock object

Value

An FLQuant object

Author(s)

The FLR Team

See Also

FLComp

Examples

```
data(ple4)
meanage(ple4)
meanageCatch(ple4)
```

meanwt — *Calculate the mean weight in stock and catch*

Description

Average weight in the stock numbers or catch-at-age.

Usage

```
meanwt(object)

meanwtCatch(object)
```

Arguments

`object` An age-structured FLStock object

Value

An FLQuant object

Author(s)

The FLR Team

See Also

FLComp

Examples

```
data(ple4)
meanwt(ple4)
meanwtCatch(ple4)
```

`metrics` *Extract simply-defined metrics from compex objects*

Description

Time series summaries of complex objects are commonly needed, for example for plotting the inputs and outputs of a class like `FLStock`. These methods allow for simple specification of those metrics by means of function calls and formulas.

Usage

```
metrics(object, metrics, ...)

## S4 method for signature 'FLComp,list'
metrics(object, metrics, ...)

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

Arguments

`object` A complex **FLR** object from which to extract time series metrics.

Value

An object, generally of class `FLQuants`.

Author(s)

The FLR Team

See Also

FLComp

Examples

```
data(ple4)
# missing
metrics(ple4)
# metrics = function
metrics(ple4, metrics=function(x) FLQuants(SSB=ssb(x), REC=rec(x),
  F=fbar(x), SSBREC=ssb(x) / rec(x)))
# metrics = formula
metrics(ple4, metrics=~ssb)
metrics(ple4, metrics=list(SSB=~ssb))
metrics(ple4, metrics=list(SBMSY=~ssb/SBMSY), FLPar(SBMSY=3.87e4))
# metrics = list
metrics(ple4, metrics=list(SSB=ssb, REC=rec, F=fbar))
metrics(ple4, metrics=list(SSB=~ssb, REC=rec, F=fbar))
```

mohnMatrix *Generate a matrix to compute Mohn's rho for a single metric*

Description

A common measure of the strength of stock assessment retrospective patterns is Mohn's rho. This function does not carry out the calculation but returns a matrix with the metrics value for the n restrospective runs, in columns, and n + 2 years, in rows.

Usage

```
mohnMatrix(stocks, metric = "fbar", ...)
```

Arguments

`stocks`	An FLStocks object from a restrospective analysis
`metric`	Metric to be computed, as a character vector or function

Value

A metrics of n + 2 x n, where n is the numbers of objects in stocks.

msy — *msy: A series of methods to extract or compute MSY-based reference points*

Description

Reference points based on equilibirum calculations of Maximum Sustainable Yield (MSY) are computed by various FLR packages. The methods' generics are defined here for convenience. Please refer to the help pages of particular methods for further details

Usage

```
msy(x, ...)

bmsy(x, ...)

sbmsy(x, ...)

fmsy(x, ...)
```

Arguments

`x`	An input object from which to extract or compute a reference point

Details

The four methods provide the following parameter estimates:

- `msy` Maximum Sustainable Yield (MSY)
- `fmsy` Fishing mortality level expected to produce on average MSY
- `bmsy` Total biomass that should produce MSY
- `sbmsy` Spawning biomass that should produce MSY

Value

A value for the requested reference point, 'FLPar'

Author(s)

The FLR Team

See Also

`FLPar`

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<-.

Usage

```
## S4 method for signature 'FLArray'
names(x)

## S4 method for signature 'FLPar'
names(x)

## S4 replacement method for signature 'FLPar,character'
names(x) <- value
```

Generic function

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

Author(s)

The FLR Team

See Also

names

Examples

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

# Contrast this with
dimnames(catch.n(ple4))
```

plot *Method plot*

Description

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

Usage

```
## S4 method for signature 'FLQuant,missing'
plot(
  x,
  xlab = "year",
  ylab = paste("data (", units(x), ")", sep = ""),
  type = "p",
  ...
)

## S4 method for signature 'FLStock,missing'
plot(x, auto.key = TRUE, ...)

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

## S4 method for signature 'FLIndex,missing'
plot(x, type = c("splom"), ...)

## S4 method for signature 'FLSR,missing'
plot(x, main = "Functional form", log.resid = FALSE, cex = 0.8)

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

Details

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

Generic function

plot(x,y)

Author(s)

The FLR Team

See Also

plot

Examples

```
data(ple4)

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

# FLStock
data(ple4sex)
plot(ple4)
plot(ple4sex)

# FLBiol
data(ple4.biol)
plot(ple4.biol)

# FLIndex
data(ple4.index)
plot(ple4.index)

# FLSR
data(nsher)
plot(nsher)

# FLPar
fpa <- FLPar(a=rnorm(100, 1, 20), b=rlnorm(100, 0.5, 0.2))
plot(fpa)
```

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.

Usage

```
## S4 method for signature 'FLQuants,formula'
predictModel(object, model, params = FLPar())

## S4 method for signature 'FLQuants,missing'
predictModel(object, params = FLPar())
```

```
## S4 method for signature 'FLQuants,character'
predictModel(object, model, params = FLPar())

## S4 method for signature 'FLQuants,function'
predictModel(object, model, params = FLPar())

## S4 method for signature 'FLQuants,list'
predictModel(object, model, params = FLPar())

## S4 method for signature 'missing,ANY'
predictModel(object, model, ...)
```

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

Examples

```
fec <- FLQuants(fec=FLQuant(rlnorm(10, 20, 5),
  dimnames=list(year=2000:2009), units='1'))
predictModel(fec, model=~fec)
predictModel(fec)
predictModel(fec, model="bevholt")
predictModel(fec, model=bevholt)
predictModel(fec, model=bevholt())
predictModel(model=rec~a*ssb, params=FLPar(a=1.234))
predictModel(model=bevholt, params=FLPar(a=1.234))
predictModel(model="bevholtss3", params=FLPar(a=1.234))
```

production *Returns the computed yearly production*

Description

Returns the computed yearly production

Usage

```
production(object, ...)

## S4 method for signature 'FLStock'
production(object, what = "ssb", ...)
```

Arguments

object	An object with biomass and catch data.
what	One of the production options: "ssb", "biomass", or "exploitation".

Details

Production can be calculated for an FLStock based on the spawning stock biomass ("ssb"), total biomass ("biomass"), or exploitation ("exploitation").

Value

The production by year, of class FLQuant.

Author(s)

Laurie Kell (Sea++), Iago Mosqueira (WMR)

Examples

```
data(ple4)
# For SSB
production(ple4, "ssb")
# For total biomass
production(ple4, "biomass")
```

propagate *Method propagate*

Description

Methods to extend objects of various FLR classes along the `iter` (6th FLQuant) dimension. Objects must generally have a single `iter` to be extended. The new iterations can be filled with copies of the existing, or remain as `NA`.

Usage

```
propagate(object, ...)

## S4 method for signature 'FLQuant'
propagate(object, iter, fill.iter = TRUE)
```

Arguments

`object`	Object to be propagated.
`fill.iter`	Should first array be copied to others? Defaults to FALSE.
`iters`	No. of iterations in output.

Generic function

propagate(object, ...)

Author(s)

The FLR Team

See Also

FLQuant

Examples

```
# An FLQuant with one iter (dim(flq)[6] == 1)
flq <- FLQuant(rnorm(80), dim=c(4,20), quant='age')

# can now be extended along the `iter` dimension, with
#' copies of the first
propagate(flq, 100)

# or without
iter(propagate(flq, 100, fill.iter=FALSE), 2)
```

properties *Returns a series of properties of the fisheries element represented by the class.*

Description

Returns a series of properties of the fisheries element represented by the class.

Usage

```
properties(object, ...)
```

Arguments

object	An object from which properties can be extracted.

Value

The correspodning properties, an FLPar.

Author(s)

Laurie Kell (Sea++), Iago Mosqueira (WMR)

quant *Method quant*

Description

Function to get or set the name of the first dimension (quant) in an object of any FLArray-based class, like FLQuant or FLCohort.

Usage

```
quant(object, ...)

## S4 method for signature 'FLArray'
quant(object)

## S4 replacement method for signature 'FLArray,character'
quant(object) <- value
```

Generic function

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

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)


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

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

quantTotals *Methods quantTotals*

Description

Methods to compute totals over selected dimensions of `FLQuant` objects 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.

Usage

```
quantTotals(x, ...)
```

Generic function

quantTotals(x)

yearTotals(x)

Author(s)

The FLR Team

See Also

FLQuant

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)
```

---

readVPAIntercatch — *Reads a single file with one year of data in VPA format as output by ICES Intercatch*

Description

Reads a single file with one year of data in VPA format as output by ICES Intercatch

Usage

```
readVPAIntercatch(file)
```

Arguments

`file`	Intercatch VPA file to load

Value

An object of class FLQuant.

residuals-FLQuant *residuals*

Description

residuals

Usage

```
## S4 method for signature 'FLQuant'
residuals(object, fit, type = "log", ...)
```

Examples

```
data(ple4)
fit <- rlnorm(1, log(catch(ple4)), 0.1)
residuals(catch(ple4), fit)
residuals(catch(ple4), fit, type="student")
rraw(catch(ple4), fit)
rlogstandard(catch(ple4), fit)
rstandard(catch(ple4), fit)
rstudent(catch(ple4), fit)
```

rnoise,numeric,FLQuant-method *Random noise with different frequencies*

Description

A noise generator

Usage

```
## S4 method for signature 'numeric,FLQuant'
rnoise(
  n = n,
  len = len,
  sd = 1,
  b = 0,
  burn = 0,
  trunc = 0,
  what = c("year", "cohort", "age"),
  seed = NA
)

## S4 method for signature 'numeric,missing'
```

```
rnoise(n = n, sd = 1, b = 0, burn = 0, trunc = 0, seed = NA)

## S4 method for signature 'numeric,FLQuant'
rlnoise(
  n = n,
  len = len,
  sd = 1,
  b = 0,
  burn = 0,
  trunc = 0,
  what = c("year", "cohort", "age"),
  seed = NA
)
```

Arguments

`n`	number of iterations
`len`	an `FLQuant`
`sd`	standard error for simulated series
`b`	autocorrelation parameter a real number in 0,1
`burn`	gets rid of 1st values i series
`trunc`	get rid of values > abs(trunc)
`what`	returns time series for year, cohort or age"
`...`	any

Value

A `FLQuant` with autocorrelation equal to B.

References

Ranta and Kaitala 2001 Proc. R. Soc. vt = b * vt-1 + s * sqrt(1 - b^2) s is a normally distributed random variable with mean = 0 b is the autocorrelation parameter

Examples

```
## Not run:
flq <- FLQuant(1:100, quant="age")
white <- rnoise(100,flq,sd=.3,b=0)
plot(white)
acf(white)

red <- rnoise(100,flq,sd=.3,b=0.7)
plot(red)
acf(red)

res <- rnoise(100,flq,sd=.3,b=0)
```

```
ggplot() +
  geom_point(aes(year,age,size=data),
    data=subset(as.data.frame(res), data>0)) +
geom_point(aes(year,age,size=-data),
            data=subset(as.data.frame(res),data<=0),colour="red")+
scale_size_area(max_size=4, guide="none")+
facet_wrap(~iter)

data(ple4)
res <- rnoise(4,m(ple4),burn=10,b=0.9,what="cohort")
ggplot()+
geom_point(aes(year,age,size= data),
          data=subset(as.data.frame(res),data>0))+
geom_point(aes(year,age,size=-data),
          data=subset(as.data.frame(res),data<=0),colour="red")+
scale_size_area(max_size=4, guide="none")+
facet_wrap(~iter)


## End(Not run)
```

roc *Receiver Operating Characteristic (ROC)*

Description

A receiver operating characteristic (ROC) curve shows the ability of a binary classifier. Here it is applied to compare two sets of values, stored as two FLQuant objects. The first is the result of aplying a logical comparison of a given state against a reference value, so it contains a binary (0, 1) label. The second, the score, contains an alternative metric that attempts to measure the absolute value of the first. The examples below compare an observation of stock status, SSB being less than a reference point, and an alternative metric, here the catch curve estimates of total mortality.

Usage

```
roc(label, ind, direction = c(">=", "<="))

auc(x = NULL, TPR = x$TPR, FPR = x$FPR)
```

Examples

```
data(ple4)
# OM 'reality' on stock status (fbar)
state <- fbar(ple4)[, ac(1960:2017)]
# Model estimates of F using catch curves
ind <- acc(catch.n(ple4)[, ac(1960:2017)])
# Compute TSS, returns data.frame
roc(state >= 0.22, ind)
# Needs ggplotFL
```

```
## Not run:
ggplot(roc(state >= 0.22, ind, direction='>='), aes(x=FPR, y=TPR)) +
  geom_line() +
  geom_abline(slope=1, intercept=0, colour="red", linetype=2)

## End(Not run)
# Computes auc using the output of roc()
with(roc(state >= 0.22, ind), auc(TPR=TPR, FPR=FPR))
auc(roc(state >= 0.22, ind))
```

ruleset *Set of verify rules for an FLR class*

Description

Returns a set of standard rules to be used by the verify method for an object of a given class.

Usage

```
ruleset(object, ...)

## S4 method for signature 'FLStock'
ruleset(object, ...)
```

Arguments

`object`	An object of any FLR class for which the method has been defined.
`...`	Names of positions in the standard list to subset.

Details

A standard minimal set of rules to check FLStock objects against using the verify method. The included rules are (with names in italics) check that:

- there are no NAs in any slot, *anyna*.
- *catch.wt*, *landings.wt*, *discards.wt* and *stock.wt* > 0.
- *mat*, *m.spwn* and *harvest.spwn* values are between 0 and 1.
- *harvest* >= 0.
- *cohorts* in the stock.n slot contain decreasing numbers, except for the plusgroup age.

Value

A named list containing the rules defined for for the class object belongs to.

Author(s)

The FLR Team

Examples

```
data(ple4)
ruleset(ple4)
# Extract single rule by name
ruleset(ple4, 'anyna')
```

---

runstest *Computes Runs Test p-values*

Description

Computes Runs Test p-values

Usage

```
runstest(fit, obs, ...)

## S4 method for signature 'FLQuants,missing'
runstest(fit, combine = TRUE)

## S4 method for signature 'FLQuants,FLQuants'
runstest(fit, obs, combine = TRUE)

## S4 method for signature 'FLQuant,FLQuant'
runstest(fit, obs, combine = TRUE)

## S4 method for signature 'FLQuant,missing'
runstest(fit, combine = TRUE)

## S4 method for signature 'numeric,numeric'
runstest(fit, obs, combine = TRUE)

## S4 method for signature 'numeric,missing'
runstest(fit, obs, combine = TRUE)
```

Arguments

fit	The result of a model fit.
obs	The observations used in the fit.
...	Extra arguments.
combine	Should ages be combined by addition, defaults to TRUE.

Value

A list with elements 'p.values' and 'pass'.

Examples

```
data(nsher)
# Compute 'runstest' for FLSR fit
runstest(fit=fitted(nsher), obs=rec(nsher))
# Example runstest by age
data(ple4)
runstest(catch.n(ple4), landings.n(ple4), combine=FALSE)
runstest(fit=FLQuants(D=residuals(catch(ple4), discards(ple4)),
  L=residuals(catch(ple4), landings(ple4))))
runstest(fit=residuals(fitted(nsher), rec(nsher)))
runstest(FLQuants(residuals(fitted(nsher), rec(nsher))))
# Returns value per iter
runstest(fit=rnorm(25, residuals(fitted(nsher), rec(nsher)), 0.2))
runstest(ssb(nsher))
runstest(rnorm(1, FLQuant(1, dimnames=list(year=1973:2021))))
runstest(rep(0.1, 10), cumsum(rnorm(10, 0.1, 0.01)))
runstest(rnorm(10, 0, 0.1))
```

---

rwalk *Generate a random walk time series from a starting point*

Description

The last year of an FLQuant object is used as atrating point to generate a time series following a random walk with drift:

$$z_t = z_{t-1} + \epsilon_t + \delta_t, t = 1, 2, ...$$

where ϵ is $\mathcal{N}(0, \sigma)$

Usage

```
rwalk(x0, end = 1, sd = 0.05, delta = 0)
```

Arguments

x0	The initial state of the random walk, ’FLQuant’.
end	The number of years or the final year of the series. numeric.
sd	The standard deviation of the random walk, numeric.
delta	The drift of the random walk.

Details

The length of the series is set by argument *end*. This is taken as a number of years, if its value is smaller than the final ’year’ of ’x0’, or as a final year if larger or of class ’character’.

Value

An ’FLQuant’ object.

Author(s)

Iago Mosqueira, WMR (2023)

See Also

FLQuant rnorm

Examples

```
data(ple4)
# Generate random walk recruitmrnt with positive drift
rwalk(rec(ple4), end=5, sd=0.08, delta=0.05)
# Use append() to add the new values at the end
append(rec(ple4), rwalk(rec(ple4), end=10, sd=0.04, delta=0))
# Use end as number of years
rwalk(rec(ple4), end=5)
# or as final year
rwalk(rec(ple4), end=2020)
```

show *Method show*

Description

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

Usage

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

## S4 method for signature 'FLArray'
print(x)
```

Details

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

Generic function

show(object)

show(object)

Author(s)

The FLR Team

See Also

FLComp

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


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

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

---

simplify *Aggregate or select along unwanted dimensions*

Description

Objects of many FLR classes might be aggregated along the "unit", "season", and/or "area" dimensions according to the type of data they contain.

Usage

```
simplify(object, ...)

## S4 method for signature 'FLStock'
simplify(
  object,
  dims = c("unit", "season", "area")[dim(object)[3:5] > 1],
  spwn.season = 1,
  rec.season = spwn.season,
  harvest = TRUE,
  weighted = FALSE
)
```

Arguments

object	A complex **FLR** object to aggregate.

Value

An object of the same class as the input.

Author(s)

The FLR Team

`slim` *Drop unnecesary 'iters'*

Description

Objects of FLR classes can vary in the length along the sixth dimension in any slot of class FLQuant. This reduces object size and memory usage. If an object has been extended fully, for example by using `propagate`, we can slim down the object by reducing any slot where all iters are identical and keeping only yhe first *iter*.

Usage

```
slim(object, ...)

## S4 method for signature 'FLComp'
slim(object, ...)
```

Arguments

`object`	A complex **FLR** object to slim down.

Details

The test for whether an slot can be slimmed is based on checking if the sum of the variance along the 6th dimensions is equal to zero.

Value

An object of the same class as the input.

Author(s)

The FLR Team

See Also

FLQuant `propagate`

Examples

```
data(ple4)
# Extend all of ple4 to 50 iters
ple4 <- propagate(ple4, 50)
# Add variability in catch.n
catch.n(ple4) <- rlnoise(50, log(catch.n(ple4)), log(catch.n(ple4))/10)
summary(ple4)
# slim object by dropping identical iters
sple4 <- slim(ple4)
summary(sple4)
```

---

split-methods *splits* x *along the* iter *dimension into the groups defined by* f.

Description

Similar to base::split, but working along the 6th, *iter*, dimension of any singular FLR object. The object is divided into as many objects as unique values in *f*, and returned as an FLlst-derived object, e.g. an FLQuants object when applied to an FLQuant.

Usage

```
## S4 method for signature 'FLQuant,vector'
split(x, f)

## S4 method for signature 'FLComp,vector'
split(x, f)
```

Arguments

x	The object to be split.
f	The vector of group names.

Value

An object of the corresponding plural class (FLQuants from FLQuant).

Author(s)

Iago Mosqueira (WMR).

Examples

```
# FROM FLQuant to FLQuants
flq <- rlnorm(20, FLQuant(seq(0.1, 0.8, length=10)), 0.2)
split(flq, c(rep(1, 5), rep(2,15)))
```

splom *Method splom*

Description

Draws a conditional scatter plot matrix.

Usage

```
## S4 method for signature 'FLPar,missing'
splom(x, data, ...)
```

Details

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

Generic function

splom(x,data)

Author(s)

The FLR Team

See Also

splom

Examples

```
flp <- FLPar(c(t(mvrnorm(500, mu=c(0, 120, 0.01, 20),
  Sigma=matrix(.7, nrow=4, ncol=4) + diag(4) * 0.3))),
  dimnames=list(params=c('a','b','c','d'), iter=1:500), units="NA")

splom(flp)
```

spread *A function to make available list elements inside a function or method*

Description

Inside a function, a call to spread() will attach to the function environment, sys.frame(), the elements in the list, or of the conversion to list of the object (e.g. named vector or FLPar), so that they be called by name. The function environment will be deleted once the function returns, so those variables won't make it to the environment from which the function was called, or further up in the call stack.

Usage

```
spread(object, FORCE = FALSE)
```

Arguments

`object`	A named list or vector whose elements are to be loaded into the calling environment.
`FORCE`	Should existing variable with matching names be redefined?

Details

By default, spread() will not overwrite variables in the function environment with the same name as any list element, unless FORCE=TRUE

Value

Invisibly the names of the variables loaded into the calling environment.

Author(s)

The FLR Team

See Also

sys.nframe

Examples

```
# EXAMPLE function
foo <- function (params) {
  a <- spread(params)
 print(a)
 x*y
}
# x and y are accesible to the internal calculation
foo(params=list(x=3.5, y=9))
```

```
# Works with FLPar
foo(params=FLPar(x=3L, y=0.99238))

# Elements in object must be named
## Not run: foo(list(3, y=0.99238))

# If a variable is missing from the spread object, function will fail
## Not run: foo(list(x=4))
# Unless the variable is already defined in the calling environment,
# in this case <environment: R_GlobalEnv>
y <- 45
foo(params=list(x=4))
```

SRModels *Stock-Recruitment models*

Description

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

Usage

```
ricker()

bevholt()

bevholtDa()

bevholtss3()

segreg()

segregDa()

geomean()

shepherd()

cushing()

rickerSV()

bevholtSV()

shepherdSV()
```

```
bevholtAR1()

rickerAR1()

segregAR1()

rickerCa()

survRec(ssf, R0, Sfrac, beta, SF0 = ssf[, 1])

bevholtsig()

mixedsrr()
```

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'.

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* to refer to the usual inputs
- 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`. If required, this function also has 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.

  $$(R_1 R_2 \ldots R_n)^{1/n} = e^{\mathbf{mean}(\log(R_1),\ldots,}$$

  $$\log(R_n))$$

- 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.

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

Author(s)

The 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.

See Also

FLSR, FLModel

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 *Calculate or return the Spawning Stock Biomass*

Description

The calculated Spawning Stock Biomass (SSB) of a fish population is returned by this method. SSB is the combined weight of all individuals in a fish stock that are capable of reproducing. In some classes this is calculated from information stored in different slots, while in others `ssb()` is simply an slot accessor. When the later is the case, the corresponding replacement method also exists.

Usage

```
ssb(object, ...)

## S4 method for signature 'FLBiol'
ssb(object, ...)
```

Arguments

`object`	Object on which `ssb` is calculated or extracted.

Details

Objects of the *FLBiol* class do not contain any information on catch or fishing mortality, so a call to `ssb()` will only correct abundances for natural mortality to the moment of spawning. The method can also take information on catches or fishing mortality and use them when calculating abundances at spawning time. An *FLQuant* named either 'catch.n', 'f', 'hr' or 'harvest' can be used. The first three are self-explanatory, while for the last units must be either 'f' or 'hr'. The quantities should refer to total yearly values, as the value in the 'spwn' slot will be used to calculate what fraction of fishing mortality to apply.

Value

An object, generally of class FLQuant.

Author(s)

The FLR Team

See Also

FLComp

Examples

```
data(ple4)
biol <- as(ple4, "FLBiol")
# SSB from FLBiol, abundances corrected only for M
ssb(biol)
# Provide catch-at-age, F or HR to correct N
ssb(biol, catch.n=catch.n(ple4))
ssb(biol, f=harvest(ple4))
ssb(biol, harvest=harvest(ple4))
ssb(biol, hr=catch.n(ple4) / stock.n(ple4))
```

---

ssb_next *Calculate next yera's SSB from survivors and Fbar*

Description

The spawning stock biomass (SSB) of the stock gets calculated from the survivors of the previous year. This provides a value for the first year after the end of the object. Weights-at-age, maturity in this extra year are calculated as averages over the last *wts.nyears*.

Usage

```
ssb_next(x, fbar = 0, wts.nyears = 3, fbar.nyears = 3)
```

Arguments

x	An FLStock object containing estimates of abundance and harvesting.
fbar	The Fbar rate assumed on the extra year. Defaults to 0.
wts.nyears	Number of years in calculation of mean weight-at-age and maturity for the extra year.
fbar.nyears	Number of years in calculation of mean selectivity, natural mortality and fraction of F abnd M before spawning for the extra year.

Details

For stocks spawning later in the year, a value for the average fishing mortality, *fbar*, expected in that year can be provided. Mortality until spawning is then calculated, with M and selectivity assumed in the extra year to be an average of the last *fbar.nyears*.

Value

An FLQuant.

Examples

```
data(ple4)
ssb_next(ple4)
# Compare with ssb()
ssb(ple4)[, ac(2014:2017)] / ssb_next(ple4)[, ac(2014:2017)]
```

`standardUnits` *Standard units of measurement for a complex class object*

Description

Returns values for the *units* of each *FLQuant* slot according to the standard adopted by the FLR Team for the supplied class.

Usage

```
standardUnits(object, ...)

## S4 method for signature 'character'
standardUnits(object, ...)

## S4 method for signature 'FLS'
standardUnits(object, ...)

## S4 method for signature 'FLBiol'
standardUnits(object, ...)
```

Arguments

`object`	for which the standard *units* are to be returned

Details

For objects derived from class *FLS*, which currently includes *FLStock* and *FLStockLen*, the adopted standard includes: 'kg' for individual weights, '1000' for number of individuals, 't' for biomass, 'f' for harvest, 'm' for natural mortality, and an empty string for proportions (spwn, mat).

For objects derived of class *FLBiol* the adopted standard units are: 'kg' for individual weights, '1000' for number of individuals, 'm' for natural mortality, and an empty string for proportions (spwn, mat).

Value

A list with the corresponding *units* value for each slot

Author(s)

The FLR Team

See Also

units-FLCore

Examples

```
stk <- FLStock(catch=FLQuant(runif(20, 2, 120)))
# FLStock object has no units
summary(stk)
# Obtain standard units for the class as a list
standardUnits(stk)
# which can then be assigned to the object
units(stk) <- standardUnits(stk)
summary(stk)
# units<- methjod also accepts a function to be called to provide units
units(stk) <- standardUnits
bio <- FLBiol(n=FLQuant(runif(50, 2, 120), dim=c(5, 10)))
# Object has no units
summary(bio)
# Obtain standard units for the class as a list
standardUnits(bio)
# which can then be assigned to the object
units(bio) <- standardUnits(bio)
summary(stk)
```

summary,FLArray-method *Method summary*

Description

Outputs a general summary of the structure and content of an fwdControl object. The method invisibly returns the data.frame shown on screen.

Usage

```
## S4 method for signature 'FLArray'
summary(object, .class = TRUE, ...)

## S4 method for signature 'FLQuantPoint'
summary(object, ...)
```

```
## S4 method for signature 'FLPar'
summary(object, title = TRUE, ...)

## S4 method for signature 'FLComp'
summary(object, ...)

## S4 method for signature 'FLQuants'
summary(object)

## S4 method for signature 'predictModel'
summary(object)

## S4 method for signature 'FLBiol'
summary(object)

## S4 method for signature 'FLModel'
summary(object, ...)

## S4 method for signature 'FLlst'
summary(object)
```

Generic function

summary(object)

Author(s)

The FLR Team

See Also

summary

Examples

```
flq <- FLQuant(rlnorm(90), dim=c(3,10), units='kg')
summary(flq)

data(ple4)
summary(ple4)

data(nsher)
summary(nsher)
```

survey *A method to generate observations of abundance at age.*

Description

A method to generate observations of abundance at age.

Usage

```
survey(object, index, ...)

## S4 method for signature 'FLStock,FLIndex'
survey(
  object,
  index,
  sel = sel.pattern(index),
  ages = dimnames(index)$age,
  timing = mean(range(index, c("startf", "endf"))),
  index.q = index@index.q,
  stability = 1
)

## S4 method for signature 'FLStock,FLIndexBiomass'
survey(
  object,
  index,
  sel = sel.pattern(index),
  ages = ac(seq(range(index, c("min")), range(index, c("max")))),
  timing = mean(range(index, c("startf", "endf"))),
  catch.wt = index@catch.wt,
  index.q = index@index.q,
  stability = 1
)

## S4 method for signature 'FLStock,missing'
survey(
  object,
  sel = catch.sel(object),
  ages = dimnames(sel)$age,
  timing = 0.5,
  index.q = 1,
  biomass = FALSE,
  stability = 1
)

## S4 method for signature 'FLStock,FLIndices'
survey(object, index, ...)
```

Arguments

`object`	The object on which to draw the observation

Value

An FLQuant for the index of abundance

Author(s)

The FLR Team

See Also

FLComp

Examples

```
data(ple4)
data(ple4.index)
# CONSTRUCT a survey from stock and index
survey(ple4, ple4.index)
# Create FLIndexBiomass
ple4.biom <- as(ple4.index, "FLIndexBiomass")
survey(ple4, ple4.biom)
data(ple4)
survey(ple4)
survey(ple4, biomass=TRUE)
```

`survivors` *Calculate the survivors of a stock to the next year.*

Description

An FLStock object containing estimates of adundance at age ('stock.n') and harvest level at age ('harvest'), is used to bring forward the population by applying the total mortality at age ('z'). No calculation is made on recruitment, so abundances for the first age will be set as 'NA', unless a value is provided.

Usage

```
survivors(object, rec = NA)
```

Arguments

`object`	An FLStock with estimated harvest and abundances
`rec`	Value for recruitment, first age abundance, 'numeric' or 'FLQuant'.'

Value

The abundances at age of the survivors, 'FLQuant'.

Examples

```
data(ple4)
stock.n(ple4[, ac(2002:2006)])
survivors(ple4[, ac(2002:2006)])
```

---

sweep,FLArray-method *Method sweep for FLCore classes*

Description

Use R's sweep method on FLCore classes

Usage

```
## S4 method for signature 'FLArray'
sweep(x, MARGIN, STATS, FUN = "-", check.margin = TRUE, ...)

## S4 method for signature 'FLPar'
sweep(x, MARGIN, STATS, FUN = "-", check.margin = TRUE, ...)
```

Details

These methods call base R sweep method on **FLCore** classes and then ensure that the returned object is of same class.

Generic function

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

Author(s)

The FLR Team

See Also

sweep

Examples

```
flq <- FLQuant(rlnorm(90), dim=c(3,10), units='kg')
# Get ratio of max value by year
sweep(flq, 2, apply(flq, 2, max), "/")
```

tail,FLQuant-method *Returns the first and last parts of an FLQuant.*

Description

Standard tail and head methods can be applied along any dimension of an FLQuant object.

Usage

```
## S4 method for signature 'FLQuant'
tail(x, n = 1, dim = 2, ...)

## S4 method for signature 'FLQuant'
head(x, n = 1, dim = 2, ...)
```

Arguments

x	The object to extract from, FLQuant.
n	The number of elements to extract, numeric.
dim	Dimension to extract from, defaults to 2, 'year'.

Value

An FLQuant with the extracted elements.

Author(s)

Iago Mosqueira (WMR)

See Also

base::tail

Examples

```
x <- FLQuant(1:10)

# Extract the last 3 years
tail(x, 3)

# Extract all but the first 3 years
tail(x, -3)

# Extract the first 3 years
head(x, 3)

# Extract all but the last 3 years
head(x, -3)
```

`trim` *Method trim*

Description

Trim FLR objects using named dimensions

Usage

```
trim(x, ...)

## S4 method for signature 'FLArray'
trim(x, ...)

## S4 method for signature 'FLComp'
trim(x, ...)

## S4 method for signature 'FLS'
trim(x, ...)

## S4 method for signature 'FLBiol'
trim(x, ...)
```

Details

Subsetting of FLR objects can be carried out with 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 to an FLQuant object of length 1 in its first dimension and with dimension name equal to 'all', values to `trim` specified for that dimension will be ignored. For example, `FLStock` objects contain slots with length=1 in their first dimension. Specifying values to trim 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 (e.g. using `computeCatch`, `computeLandings`, `computeDiscards` and `computeStock`) if their quant-structured counterparts are modified along the first dimension.

Generic function

trim(x)

Author(s)

The FLR Team

See Also

FLQuant, FLStock, FLCohort, FLIndex

Examples

```
flq <- FLQuant(rnorm(90), dimnames=list(age=1:10, year=2000:2016))

trim(flq, year=2000:2005)
# which is equivalent to
window(flq, start=2000, end=2005)

trim(flq, year=2000:2005, age=1:2)


# Now on an FLStock
data(ple4)
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-FLCore *Method units*

Description

units attribute for FLQuant and FLArray-derived objects

Usage

```
## S4 method for signature 'FLArray'
units(x)

## S4 replacement method for signature 'FLArray,character'
units(x) <- value

setunits(x, value)

## S4 method for signature 'FLPar'
units(x)

## S4 replacement method for signature 'FLPar,character'
units(x) <- value

## S4 method for signature 'FLComp'
units(x)
```

```
## S4 replacement method for signature 'FLComp,list'
units(x) <- value

## S4 replacement method for signature 'FLComp,character'
units(x) <- value

## S4 replacement method for signature 'FLComp,function'
units(x) <- value
```

Details

Objects of FLArray-based classes (e.g. FLQuant) contain a 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.

The complete set of *units* for a complex object can be obtained as a named *list*.

Assignment of *units* to the *FLQuant* slots of a complex object can be carried out passing a named *list* or *character* vector containing the units for the slots to be modified.

Generic function

units(x)

units<-(x,value)

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')

data(ple4)
units(ple4) <- list(harvest="hr")
units(ple4) <- c(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)

uomUnits(unit = missing)
```

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
unit	A character vector for one or more units to be compared with those known to uom.

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

The list of units known to `uom` is stored internally but can be queried by calling `uomUnits()` with no arguments. If a character vector is provided, a logical is returned telling whether the string is included or not in that table.

Value

`uom` returns a string with the corresponding units of measurement, or a character vector, showing the operation carried out, when units are not known to `uom` or not compatible, e.g. "100 * d".

`uomUnits` returns TRUE or FALSE if `unit` is given, otherwise a character vector with all units known to `uom`.

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')

# Check if units are known
uomUnits('kg')
uomUnits('kell')
```

---

uomTable *Table for conversions and operations between units of measurement*

Description

- uom defaults to NA unless defined below.
- unit +/- itself, returns the same unit (e.g. kg + kg = kg)
- numeric unit * 1 returns same unit (e.g. 1e4 * 1 = 1e4)
- numeric unit * numeric unit returns product (e.g. 10 * 100 = 1000)
- unit / unit returns "" (e.g. 100 / 100 = "")
- numeric unit / smaller numeric unit returns division (e.g. 100 / 10 = 10)
- 100 times kg returns t
- numeric unit * 'kg' returns the product in tonnes (e.g. kg * 1e4 = t * 10)
- units with divisions are parsed (e.g. days/boat * boat = days)
-
-

Format

An object of class array

`upperlower` *Extract and modify the* lower *and* upper *FLModel attibutes.*

Description

Description: Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Usage

```
lower(object, ...)

## S4 method for signature 'FLModel'
lower(object)

## S4 method for signature 'FLModel'
upper(object)
```

Arguments

`object`	Object to extract from or modify
`...`	Other arguments
`value`	New value

Details

Details: Aliquam sagittis feugiat felis eget consequat.

Value

RETURN Lorem ipsum dolor sit amet

Author(s)

The FLR Team

See Also

`FLModel`

verify *Verify FLR objects*

Description

Verifies the content of FLR objects according to a set of rules

Usage

```
verify(object, ...)

## S4 method for signature 'FLComp'
verify(object, ..., report = TRUE)

## S4 method for signature 'FLStock'
verify(object, rules = ruleset(object), ..., report = TRUE)
```

Arguments

`object`	An object of any FLR class for which the method has been defined
`...`	Additional rules to be tested, as a formula or list. See details
`report`	Should the standard data.frame report be output (if TRUE) or a single logical value for all tests?
`rules`	Basic set of rules for a given class, as returned by ruleset().

Details

Classes' validity functions generally check the structure and dimensions of objects and their component slots. But some checks on the data content of objects is often required. The various verify methods implement both a system to create *rules* that an object is expected to pass, and a minimum standard set of rules for each defined class

The data.frame output by the method when `report=TRUE` contains one row per rule and the following columns:

- *name*, the rule name
- *items*, number of comparisons carried out
- *passes*, number of *TRUE* values
- *fails*, number of *FALSE* values
- *NAs*, number of logical NAs
- *valid*, are all values *TRUE*?
- *rule*, the expression being evaluated

Additional rules can be specify in a call to *verify*, in one of two forms. Simple rules can be defined as a formula involving methods defined for the class. A rule such as `highm = ~ m < 2` will check if values in the *m* slot are higher than 2 and return a logical *FLQuant*.

Some rules cannot simply use existing methods or functions, for example those operating on all slots of the object, or requiring additional computations. In this case, the argument to *verify* can be a list, with an element named *rule* of class *formula* and where test is defined. The test then calls for a new function, defined as another element of the list, and which will be used by verify when evaluating the set of rules. See below for examples.

A set of rules has been defined for the *FLStock* class, available by calling the ruleset method. The verify method for *FLStock* will by default evaluate those rules, as well as any other defined in the call.

Value

A data.frame with the results of applying those rules, or a single logical value, if report=FALSE

Author(s)

The FLR Team

See Also

`ruleset`

Examples

```
# Verifying a new rule for an FLSR object
data(nsher)
# rule: are all recruitment values greater than 0?
verify(nsher, rec=~rec > 0)

# Define rule calling its own function
data(ple4)
# rule: ssb is less
verify(ple4, ssbstock = ~ssb < stock)
data(ple4)
# verify for the standard set of rules for FLStock
verify(ple4)
# verify a single rule from set
verify(ple4, rules=ruleset(ple4, 'anyna'), report=FALSE)

# add own rule to set
verify(ple4, m = ~m >=0)
```

vonbert *Growth models*

Description

Growth models

ivonbert

gompertz

richards

Usage

```
vonbert(linf, k, t0, age)

ivonbert(linf, k, t0, len)

gompertz(linf, a, k, age)

richards(linf, k, b, m, age)
```

Examples

```
data(ple4)
vonbert

vonbert(linf=35, k=0.352, t0=-0.26, age=1:14)
ivonbert(35, 0.352, -0.26, 1:34)
gompertz(linf=179.13, k=0.4088, a=1.7268, age=1:12)
richards(linf=178.63, k=0.424, b=-7.185, m=2880.4, age=1:12)
```

weighted.mean,FLQuants,FLQuants-method *Weighted means along a FLQuants.*

Description

Facilitates the calculation of weighted means across a FLQuants object.

Usage

```
## S4 method for signature 'FLQuants,FLQuants'
weighted.mean(x, w)
```

Arguments

`x`	Values to be averaged, as an object of class `FLQuants`.
`w`	weights to be used, as an object of class `FLQuants`.

Details

An object of class FLQuants containing elements over which an average is to computed, is combined with another one, of the same length, containing values to be used as weights. The overall weighted mean is calculated by computing the product of each element to its corresponding weight, and dividing by the sum of all weights. NAs in the value elements are substituted for zeroes, so do not influence the mean.

Value

A single `FLQuant` object.

Author(s)

The FLR Team

See Also

FLCore::FLQuants stats::weighted.mean

Examples

```
data(ple4)
# Weighted mean of landings and discards weights-at-age
weighted.mean(FLQuants(L=landings.wt(ple4), D=discards.wt(ple4)),
  FLQuants(L=landings.n(ple4), D=discards.n(ple4)))
```

`wireframe` *Method wireframe*

Description

3D plot for 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`

Details

Method to plot 3D representations of FLQuant objects

Value

a `wireframe` plot

Examples

```
data(ple4)
wireframe(data~age+year, data=harvest(ple4))
```

`yearSample` *Samples along the year dimension*

Description

A resample from an FLQuant object along the 'year' dimension is returned. The 'year' dimnames of the output object can be specified, although that is not needed if the resample is to be assigned in a slot.

Usage

```
yearSample(x, size = length(years), years, replace = TRUE, prob = NULL)
```

Arguments

`x`	An FLQuant object.
`size`	Number of samples (years), non-negative integer.
`years`	Optional vector to set as 'year' dimnames in output.
`replace`	should sampling be with replacement? Defaults to TRUE.
`prob`	a vector of probability weights.

Value

RETURN Description, class

Author(s)

Iago Mosqueira (WMR)

See Also

FLQuant `sample()`

Examples

```
data(ple4)
# Take 20 samples of recent recruitment
yearSample(rec(ple4)[, ac(2013:2017)], 20)
# Providing 'years' sets the output object dimnames
yearSample(rec(ple4)[, ac(2013:2017)], 20, year=2000:2019)
```

z *Total mortality z*

Description

Returns the calculation of total mortality, *z*, usually as the sum of fishing mortality, *f*, and natural mortality, *m*.

Usage

```
z(object, ...)

## S4 method for signature 'FLS'
z(object, ...)
```

Arguments

`object`	Object to calculate on.
`...`	Any extra arguments.

Value

An object of the corresponding class, usually *FLQuant*.

Author(s)

The FLR Team

See Also

`FLQuant`

Examples

```
data(ple4)

z(ple4)
```

%+% FLQuant arithmetic operators that extend objects

Description

Arithmetic operations between two FLQuant objects using the standars operators (`+`, `-`, `*`, `/`, `^`, see Arith) need all dimensions in both objects to match. This requirement is relaxed by using the percent version of those five operators: `%+%`, `%-%`, `%*%`, `%/%` and `%^%`.

Usage

```
e1 %+% e2

x %-% y

x %^% y

## S4 method for signature 'FLQuant,FLQuant'
x %*% y

## S4 method for signature 'FLQuant,FLQuant'
e1 %/% e2

## S4 method for signature 'FLQuant,FLQuant'
e1 %+% e2

## S4 method for signature 'FLQuant,FLQuant'
x %-% y

## S4 method for signature 'FLQuant,FLQuant'
x %^% y

## S4 method for signature 'FLPar,FLQuant'
x %*% y

## S4 method for signature 'FLPar,FLQuant'
e1 %/% e2

## S4 method for signature 'FLPar,FLQuant'
e1 %+% e2

## S4 method for signature 'FLPar,FLQuant'
x %-% y

## S4 method for signature 'FLPar,FLQuant'
x %^% y
```

```
## S4 method for signature 'FLQuant,FLPar'
x %*% y

## S4 method for signature 'FLQuant,FLPar'
e1 %/% e2

## S4 method for signature 'FLQuant,FLPar'
e1 %+% e2

## S4 method for signature 'FLQuant,FLPar'
x %-% y

## S4 method for signature 'FLQuant,FLPar'
x %^% y

## S4 method for signature 'FLPar,FLPar'
x %*% y

## S4 method for signature 'FLPar,FLPar'
e1 %+% e2

## S4 method for signature 'FLPar,FLPar'
x %-% y

## S4 method for signature 'FLPar,FLPar'
e1 %/% e2

## S4 method for signature 'FLPar,FLPar'
x %^% y

## S4 method for signature 'FLQuants,FLPar'
e1 / e2

## S4 method for signature 'FLQuants,FLPar'
e1 * e2

## S4 method for signature 'FLQuants,FLPars'
e1 / e2

## S4 method for signature 'FLQuants,FLPars'
e1 * e2

## S4 method for signature 'FLQuants,FLQuants'
e1 / e2

## S4 method for signature 'FLQuants,FLQuants'
e1 * e2
```

```
## S4 method for signature 'FLQuants,FLQuants'
e1 + e2

## S4 method for signature 'FLQuants,FLQuants'
e1 - e2
```

Details

If any of the objects is of length one in a dimensions where the other is longer, the dimensions will be extended and the element-by-element operation then conducted. Dimensions and dimnames of the output will be those of the larger object. See the examples to observe their behaviour.

Please note that this behaviour is already present on the Arith methods for FLArray-derived classes but only on the 6th, `iter`, dimension.

The original use of the `%*%` operator, as vector product, is not available for FLQuant objects, but can be applied to the array inside them, as in the example below.

Methods for operations between an FLQuant and an FLPar object will match dimensions by names of dimnames, regardless of position.

Generic function

x %+% y, x %-% y, x %*% y, e1 %/% e2, x %^% y

Author(s)

The FLR Team

See Also

`FLQuant`, `matmult`

Examples

```
a <- FLQuant(2, dim=c(3,3,2))
b <- FLQuant(3, dim=c(3,3,1))

# This should fail
## Not run:  a * b

a %*% b
a %+% b
# To use base's %*% vector product, apply it to a matrix from @.Data
b@.Data[,,,,,] %*% 1:3
# or
b[,,drop=TRUE] %*% 1:3

# FLPar vs. FLQuant works by dimnames' names
flp <- FLPar(2, dimnames=list(params='a', year=2000:2005, iter=1))
flq <- FLQuant(3, dimnames=list(year=2000:2005))
flp %*% flq
```

```
# Divide each FLQuants element by a 'param' in FLPar, e.g. time series
# divide by reference points
FLQuants(SSB=FLQuant(2303), F=FLQuant(0.8)) / FLPar(SSB=1560, F=0.4)

# Product of each FLQuants element by a 'param' in FLPar
FLQuants(SSB=FLQuant(2303), F=FLQuant(0.8)) * FLPar(SSB=1560, F=0.4)
# Divide each FLQuants element by each in FLPars
FLQuants(A=FLQuant(2303), B=FLQuant(1287)) /
  FLPars(A=FLPar(SBMSY=1560), B=FLPar(SBMSY=1000))
# Divide each FLQuants element by each in FLPars
FLQuants(A=FLQuant(2303), B=FLQuant(1287)) *
  FLPars(A=FLPar(SBMSY=1560), B=FLPar(SBMSY=1000))
# Divide each FLQuants element by each in FLPars
FLQuants(A=FLQuant(300), B=FLQuant(200)) /
  FLQuants(A=FLQuant(3), B=FLQuant(2))
# Divide each FLQuants element by each in FLPars
FLQuants(A=FLQuant(100), B=FLQuant(200)) *
  FLQuants(A=FLQuant(3), B=FLQuant(2))
# Divide each FLQuants element by each in FLPars
FLQuants(A=FLQuant(100), B=FLQuant(200)) *
  FLQuants(A=FLQuant(3), B=FLQuant(2))
# Divide each FLQuants element by each in FLPars
FLQuants(A=FLQuant(100), B=FLQuant(200)) *
  FLQuants(A=FLQuant(3), B=FLQuant(2))
```

Index