Package: FLa4a (via r-universe)

August 28, 2024

Type Package

Title A Simple and Robust Statistical Catch at Age Model

Version 1.8.3.9001

Description A simple and robust statistical Catch at Age model that is specifically designed for stocks with intermediate levels of data quantity and quality.

License EUPL

Imports methods, lattice, Matrix, copula, coda, grid, gridExtra, latticeExtra, mgcv

Depends R (>= 4.0), FLCore (>= 2.6.15), triangle,

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

Suggests knitr, formatR, XML, reshape2, testthat

LazyLoad yes

LazyData yes

VignetteBuilder knitr

Collate 'FLCompMethods.R' 'FLModelSimMethods.R' 'a4aM-class.R' 'a4aM-methods.R' 'a4aGr-class.R' 'a4aGr-methods.R' 'a4aStkParams-class.R' 'submodel-class.R' 'submodels-class.R' 'SCAPars-class.R' 'SCAMCMC-class.R' 'a4aFit-class.R' 'a4aFitSA-class.R' 'a4aFitSAs-class.R' 'a4aFitMCMC-class.R' 'a4aFitresiduals-class.R' 'coef-methods.R' 'vcov-methods.R' 'predict-methods.R' 'simulate-methods.R' 'addition-methods.R' 'internal.R' 'data.R' 'redfish-data.R' 'setupModel.R' 'fittingFunctions.R' 'l2a-methods.R' 'ma-methods.R' 'utilities.R' 'gen-methods.R' 'southern_hake-data.R' 'a4amse-sa.R' 'a4aFitCatchDiagn-class.R'

RoxygenNote 7.2.1

Encoding UTF-8

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

RemoteUrl https://github.com/flr/FLa4a

RemoteRef HEAD

RemoteSha 4bbe0d03bc1faacfbc341defd584adad3d96753b

Contents

`*` ** methods*

Description

Update `FLStock` and `FLIndex` objects with simulations from stock assessment fits.

Usage

```
## S4 method for signature 'FLStock,a4aFitSA'
e1 * e2

## S4 method for signature 'FLStock,SCAPars'
e1 * e2

## S4 method for signature 'FLIndices,a4aFitSA'
e1 * e2

## S4 method for signature 'FLIndices,SCAPars'
e1 * e2
```

Arguments

e1	the original FLStock or FLIndex object
e2	a a4aFit object from where the new FLStock or FLIndex slots will be extracted.

---

a4aFit-class *S4 class* a4aFit

Description

The a4aFit class was built to store the a4a stock assessment fits.

Usage

```
a4aFit(...)

a4aFit(...)

clock(object, ...)

## S4 method for signature 'a4aFit'
clock(object)

fitSumm(object, ...)

## S4 method for signature 'a4aFit'
fitSumm(object)

## S4 method for signature 'a4aFit'
stock.n(object)

## S4 method for signature 'a4aFit,ANY'
harvest(object)

## S4 method for signature 'a4aFit'
catch.n(object)

## S4 method for signature 'a4aFit'
index(object)

## S4 method for signature 'a4aFit'
show(object)

## S4 method for signature 'a4aFit'
logLik(object, ...)

## S4 method for signature 'a4aFit'
```

```
iter(obj, it)

## S4 method for signature 'a4aFit'
computeCatchDiagnostics(object, stock, ...)
```

Arguments

`...`	additional argument list that might never be used
`object`	object of relevant class (see signature of method)
`obj`	the object to be subset
`it`	iteration to be extracted

Slots

call The function call

clock Information on call duration

fitSumm Fit summary

stock.n Estimates of stock numbers-at-age

harvest Estimates of fishing mortality at age

catch.n Estimates of catch numbers-at-age

index Estimates of survey or CPUE indices-at-age

Accessors

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

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

Constructor

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

Examples

```
data(ple4)
data(ple4.index)

obj <- sca(stock=ple4, indices=FLIndices(ple4.index))
obj

slotNames(obj)
clock(obj)
fitSumm(obj)
```

```
flq <- stock.n(obj)
is(flq)
flq <- index(obj)
is(flq)

logLik(obj)
AIC(obj)
BIC(obj)
```

---

a4aFitCatchDiagn-class *S4 class* a4aFitCatchDiagn

Description

The a4aFitCatchDiagn class extends FLQuants to store information to run diagnostics on aggregated catch estimated by the a4a stock assessment fit.

Usage

```
computeCatchDiagnostics(object, ...)
```

Arguments

object	object of relevant class (see signature of method)
...	additional argument list that might never be used
stock	FLStock object used to fit the model
indices	FLIndices object used to fit the model

Examples

```
data(ple4)
data(ple4.index)
obj <- sca(stock=ple4, indices=FLIndices(ple4.index))
flqs <- residuals(obj, ple4, FLIndices(idx=ple4.index))
```

a4aFitMCMC-class *S4 class* a4aFitMCMC

Description

The a4aFitMCMC class extends a4aFitSA to store information about the MCMC run.

Usage

```
a4aFitMCMC(...)

a4aFitMCMC(...)

## S4 method for signature 'a4aFitMCMC'
a4aFitSA(object, ...)

## S4 method for signature 'a4aFitMCMC'
a4aFit(object, ...)

as.mcmc(x, ...)

## S4 method for signature 'a4aFitMCMC'
as.mcmc(x, ...)

burnin(object, ...)

## S4 method for signature 'a4aFitMCMC'
burnin(object, burnin)
```

Arguments

...	additional argument list that might never be used
object	object of relevant class (see signature of method)
x	an object to be coerced into mcmc
burnin	a numeric with the number of iterations to be removed

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.

call The function call

clock Information on call duration

fitSumm Fit summary

stock.n Estimates of stock numbers-at-age

harvest Estimates of fishing mortality at age

catch.n Estimates of catch numbers-at-age

index Estimates of survey or CPUE indices-at-age

mcmc An object of class `SCAMCMC` with information about the MCMC run

Accessors

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

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

Constructor

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

Examples

```
data(ple4)
data(ple4.index)

obj <- sca(stock=ple4, indices=FLIndices(ple4.index), fit="assessment")
obj

slotNames(obj)
clock(obj)
fitSumm(obj)

flq <- stock.n(obj)
is(flq)
flq <- index(obj)
is(flq)

logLik(obj)
AIC(obj)
BIC(obj)

is(pars(obj))
```

a4aFitResiduals-class *S4 class* a4aFitResiduals

Description

The a4aFitResiduals class extends FLQuants to store residuals of the a4a stock assessment fit. By default, these should be log residuals of catches and indices.

Usage

```
## S4 method for signature 'a4aFit'
residuals(object, stock, indices, ...)
```

Arguments

object	object of relevant class (see signature of method)
stock	FLStock object used to fit the model
indices	FLIndices object used to fit the model
...	additional argument list that might never be used

Examples

```
data(ple4)
data(ple4.index)
obj <- sca(stock=ple4, indices=FLIndices(ple4.index))
flqs <- residuals(obj, ple4, FLIndices(idx=ple4.index))
```

a4aFitSA-class *S4 class* a4aFitSA

Description

The a4aFitSA class extends a4aFit to store information about the parameters of the model.

Usage

```
a4aFitSA(...)

a4aFitSA(...)

## S4 method for signature 'a4aFitSA'
a4aFit(object, ...)

pars(object)
```

```
## S4 method for signature 'a4aFitSA'
pars(object)

## S4 method for signature 'a4aFitSA'
m(object)

## S4 method for signature 'a4aFitSA'
wt(object)

## S4 method for signature 'a4aFitSA'
qmodel(object)

## S4 method for signature 'a4aFitSA'
fmodel(object)

## S4 method for signature 'a4aFitSA'
srmodel(object)

## S4 method for signature 'a4aFitSA'
n1model(object)

## S4 method for signature 'a4aFitSA'
vmodel(object)

## S4 method for signature 'a4aFitSA'
stkmodel(object)

## S4 method for signature 'a4aFitSA'
show(object)

## S4 method for signature 'a4aFitSA'
submodels(object, ...)

## S4 method for signature 'a4aFitSA'
iter(obj, it)

a4aFitSAs(object, ...)

## S4 method for signature 'list'
a4aFitSAs(object, ...)

## S4 method for signature 'a4aFitSA'
a4aFitSAs(object, ...)

## S4 method for signature 'missing'
a4aFitSAs(object, ...)
```

Arguments

...	additional argument list that might never be used
object	object of relevant class (see signature of method)
obj	the object to be subset
it	iteration to be extracted

Slots

call The function call

clock Information on call duration

fitSumm Fit summary

stock.n Estimates of stock numbers-at-age

harvest Estimates of fishing mortality at age

catch.n Estimates of catch numbers-at-age

index Estimates of survey or CPUE indices-at-age

pars an object of class SCAPars with information about model parameters

Accessors

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

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

Constructor

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

Examples

```
data(ple4)
data(ple4.index)

obj <- sca(stock=ple4, indices=FLIndices(ple4.index), fit="assessment")
obj

slotNames(obj)
clock(obj)
fitSumm(obj)

flq <- stock.n(obj)
is(flq)
flq <- index(obj)
is(flq)
```

```
logLik(obj)
AIC(obj)
BIC(obj)

is(pars(obj))
```

---

a4aGr *Individual growth class*

Description

Class definition (slots), constructors, accessors, replacement (when relevant) and common methods.

Usage

```
a4aGr(object, ...)

## S4 method for signature 'missing'
a4aGr(object, ...)

grMod(object, ...)

## S4 method for signature 'a4aGr'
grMod(object)

grMod(object) <- value

## S4 replacement method for signature 'a4aGr,formula'
grMod(object) <- value

grInvMod(object, ...)

## S4 method for signature 'a4aGr'
grInvMod(object)

grInvMod(object) <- value

## S4 replacement method for signature 'a4aGr,formula'
grInvMod(object) <- value

## S4 method for signature 'a4aGr'
params(object)

## S4 replacement method for signature 'a4aGr,FLPar'
params(object) <- value
```

```
## S4 method for signature 'a4aGr'
distr(object)

## S4 replacement method for signature 'a4aGr,character'
distr(object) <- value

## S4 method for signature 'a4aGr'
vcov(object)

## S4 replacement method for signature 'a4aGr,numeric'
vcov(object) <- value
```

Arguments

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

Slot

grMod the formula for the growth model, *e.g.* von Bertallanffy

grInvMod the formula for the inverse of the growth model, having length as the independent variable

params an FLPar object with the parameters of the model; must match the equations in the models

vcov an array with the variance covariance matrix of the parameters

distr a character with the parameters' statistical distribution; it must match a known distribution for R (*e.g.* "norm" for gaussian), so that rnorm can be called

Accessors

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

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

Constructor

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

Examples

```
mm <- matrix(NA, ncol=3, nrow=3)
diag(mm) <- c(50, 0.001,0.001)
mm[upper.tri(mm)] <- mm[lower.tri(mm)] <- c(0.1,0.01,0.00004)
md <- ~linf*(1-exp(-k*(t-t0)))
```

```
imd <- ~t0-1/k*log(1-len/linf)
prs <- FLPar(linf=58.5, k=0.086, t0=0.001, units=c("cm","yr^-1","yr"))
vbObj <- a4aGr(grMod=md, grInvMod=imd, params=prs, vcov=mm, distr="norm")
```

---

a4aInternal *Stock assessment model advanced method*

Description

The advanced user interface to the a4a fitting routine.

Usage

```
a4aInternal(
  stock,
  indices,
  fmodel = defaultFmod(stock),
  qmodel = defaultQmod(indices),
  srmodel = defaultSRmod(stock),
  n1model = defaultN1mod(stock),
  vmodel = defaultVmod(stock, indices),
  covar = missing,
  wkdir = missing,
  verbose = FALSE,
  fit = "assessment",
  center = TRUE,
  mcmc = missing
)
```

Arguments

stock	an FLStock object containing catch and stock information
indices	an FLIndices object containing survey indices
fmodel	a formula object depicting the model for log fishing mortality at age
qmodel	a list of formula objects depicting the models for log survey catchability at age
srmodel	a formula object depicting the model for log recruitment
n1model	a formula object depicting the model for the first year of catch data
vmodel	a list of formula objects depicting the models for log survey and log fishing mortality variance
covar	a list with covariates
wkdir	used to set a working directory for the admb optimiser. If wkdir is set all admb files are saved to this folder otherwise they are deleted.
verbose	if true admb fitting information is printed to the screen
fit	character with type of fit: 'MP' or 'assessment'; the former doesn't require the hessian to be computed, while the latter does.

center	logical specifying whether data is centered before estimating or not
mcmc	SCAMCMC specifying parameters for the ADMB MCMC run, check ADMB manual for detailed description

Value

an a4aFit object if fit is "MP" or an a4aFitSA if fit is "assessment"

---

a4aM *Natural mortality class*

Description

Class definition (slots), constructors, accessors, replacement (when relevant) and common methods.

Usage

```
a4aM(object, ...)

## S4 method for signature 'missing'
a4aM(object, ...)

## S4 method for signature 'a4aM'
show(object)

shape(object, ...)

## S4 method for signature 'a4aM'
shape(object)

shape(object) <- value

## S4 replacement method for signature 'a4aM'
shape(object) <- value

level(object, ...)

## S4 method for signature 'a4aM'
level(object)

level(object) <- value

## S4 replacement method for signature 'a4aM'
level(object) <- value

trend(object, ...)
```

```
## S4 method for signature 'a4aM'
trend(object)

trend(object) <- value

## S4 replacement method for signature 'a4aM'
trend(object) <- value
```

Arguments

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

Slot

shape the shape of M by age

level the mean level of M over a range of ages, which will be used to scale the shape

trend the yearly trend in M

Accessors

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

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

Constructor

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

Examples

```
mod1 <- FLModelSim(model=~exp(-age-0.5))
mod2 <- FLModelSim(model=~1.5*k, params=FLPar(k=0.4))
m1 <- a4aM(shape=mod1, level=mod2)
```

a4aStkParams *Stock parameters class*

Description

Class definition (slots), constructors, accessors, replacement (when relevant) and common methods.

Usage

```
a4aStkParams(object, ...)

## S4 method for signature 'missing'
a4aStkParams(object, ...)

## S4 method for signature 'a4aStkParams'
m(object)

## S4 method for signature 'a4aStkParams'
wt(object)

## S4 method for signature 'a4aStkParams'
mat(object)

fMod(object, ...)

## S4 method for signature 'a4aStkParams'
fMod(object)

fMod(object) <- value

## S4 replacement method for signature 'a4aStkParams,formula'
fMod(object) <- value

n1Mod(object, ...)

## S4 method for signature 'a4aStkParams'
n1Mod(object)

n1Mod(object) <- value

## S4 replacement method for signature 'a4aStkParams,formula'
n1Mod(object) <- value

srMod(object, ...)

## S4 method for signature 'a4aStkParams'
srMod(object)
```

```
srMod(object) <- value

## S4 replacement method for signature 'a4aStkParams,formula'
srMod(object) <- value

## S4 method for signature 'a4aStkParams'
params(object)

## S4 replacement method for signature 'a4aStkParams,FLPar'
params(object) <- value

coefficients(object, ...)

## S4 method for signature 'a4aStkParams'
coefficients(object)

coefficients(object) <- value

## S4 replacement method for signature 'a4aStkParams,FLPar'
coefficients(object) <- value

## S4 method for signature 'a4aStkParams'
distr(object)

## S4 replacement method for signature 'a4aStkParams,character'
distr(object) <- value

## S4 method for signature 'a4aStkParams'
vcov(object)

## S4 replacement method for signature 'a4aStkParams,array'
vcov(object) <- value

## S4 method for signature 'a4aStkParams'
propagate(object, iter, fill.iter = TRUE)

## S4 method for signature 'a4aStkParams'
iter(obj, it)
```

Arguments

object	object of relevant class (see signature of method)
...	additional argument list that might never be used
value	the new object
iter	the number of iterations to create
fill.iter	should the new iterations be filled with values (TRUE) or NAs (FALSE)

obj	the object to be subset
it	iteration to be extracted

Slot

fMod F submodel formula

n1Mod first year N formula

srMod stock-recruitment submodel formula

params FLPar with parameters

vcov array with variance-covariance

centering centering values numeric

distr statistical distribution character

m natural mortality FLQuant

units data units character

Accessors

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

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

Constructor

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

---

addition + *methods*

Description

Update FLStock and FLIndex objects with stock assessment results.

Usage

```
## S4 method for signature 'FLStock,a4aFit'
e1 + e2

## S4 method for signature 'FLIndices,a4aFit'
e1 + e2
```

Arguments

e1	the original FLStock or FLIndex object
e2	a a4aFit object from where the new FLStock or FLIndex slots will be extracted.

Details

If both objects have the same number of iterations, the FLStock slots will be replaced by the a4aFit slots, in the case of 1 iter, or a4aFitSA slots, in the case of n iters. If one of the objects has 1 iter and the other n, the method will simulate using the fit results from the a4aFitSA object to update the slots of the FLStock object.

---

assorted methods *Assorted methods needed by FLa4a*

Description

Assorted methods needed by FLa4a

Assorted methods needed by FLa4a

Usage

```
getYidx(object, ...)

## S4 method for signature 'FLQuant'
getYidx(object, year)

is.empty(object)

niters(object, ...)

## S4 method for signature 'FLModelSim'
niters(object)

## S4 method for signature 'a4aGr'
niters(object)

## S4 method for signature 'a4aStkParams'
dims(obj)

replaceZeros(object, ...)

## S4 method for signature 'FLQuant'
replaceZeros(object, fraction = 0.25)

## S4 method for signature 'FLStock'
replaceZeros(object, fraction = 0.25)
```

```
## S4 method for signature 'FLIndex'
replaceZeros(object, fraction = 0.25)

## S4 method for signature 'FLIndices'
replaceZeros(object, fraction = 0.25)
```

Arguments

object	object of relevant class (see signature of method)
...	additional argument list that might never be used
year	numeric with year to be extracted
obj	an object
fraction	numeric with fraction of minimum to replace zeros

getYidx

Gets an FLQuant's numeric id for a vector of "years". For internal use and not very interesting for users. It takes an FLQuant object and vector of years and returns a numeric vector that can be used to subset the FLQuant.

is.empty

Method is.empty checks if an object is empty. It takes any object and returns a logical, TRUE, if the object is of length 0.

niters

Compute number of iterations. Takes an object of any FLR class and returns a numeric.

dims

Extracts the dims of the parameters.

replaceZeros

Replaces observations of 0 by a fraction of the minimum observed. It takes an FLQuant object and numeric of min fraction and returns a FLQuant with zeros replaced to be added to the FLStock or FLIndex objects.

Examples

```
#Example use of getYidx:
data(ple4)
flq <- catch(ple4)
getYidx(flq, 2000:2004)
flq[, getYidx(flq, 2000:2004)]
#Example use of is.empty:
is.empty(list())
```

```
is.empty(list(a=2))
#Example use of niters:
mm <- matrix(NA, ncol=3, nrow=3)
diag(mm) <- c(50, 0.001,0.001)
mm[upper.tri(mm)] <- mm[lower.tri(mm)] <- c(0.1,0.01,0.00004)
md <- ~linf*(1-exp(-k*(t-t0)))
imd <- ~t0-1/k*log(1-len/linf)
prs <- FLPar(linf=58.5, k=0.086, t0=0.001, units=c("cm","yr^-1","yr"))
vbObj <- a4aGr(grMod=md, grInvMod=imd, params=prs, vcov=mm, distr="norm")
# Generate 100 sample sets
vbObj <- mvrnorm(100,vbObj)
niters(vbObj)
#Example use of dims:
dims(FLPar())
#Example use of getYidx:
data(ple4)
flq <- catch(ple4)
flq <- replaceZeros(flq, 0.25)
catch(ple4) <- flq
```

`breakpts` *Breakpoints*

Description

Method to set breakpoints in submodels

Usage

```
breakpts(var, ...)

## S4 method for signature 'numeric'
breakpts(var, breaks, ...)
```

Arguments

`var`	a `numeric` object that defines the variable to be "broken"
`...`	additional argument list that might never be used
`breaks`	a `numeric` object that defines the breakpoints

Value

a `factor` with levels according to the defined breaks

bubble plot of residuals

Bubbles plot of standardized log residuals

Description

Method to produce bubble plots of standardized residuals

Usage

```
## S4 method for signature 'a4aFitResiduals,missing'
bubbles(x, data = missing, ...)
```

Arguments

x	an a4aFitResiduals object with the standardized residuals
data	ignored
...	additional argument list that might never be used

Value

a bubbles plot with stardardized log residuals

Examples

```
data(ple4)
data(ple4.index)
obj <- sca(ple4, FLIndices(ple4.index))
flqs <- residuals(obj, ple4, FLIndices(idx=ple4.index))
bubbles(flqs)
```

collapseSeasons *Collapse seasons*

Description

Method to collapse seasons of FLStock objects. M and catch-at-age are summed while mean weights at age, maturity at age and mortalities before spawning are averaged.

Usage

```
collapseSeasons(stock)
```

Arguments

stock	an FLStock object

Value

a FLStock object

`defaultSubModels` *Default sub-models*

Description

Methods to create formulas for sub-models. The sub-models are set automagically using defaults.

Usage

```
defaultFmod(stock, dfm = c(0.5, 0.7))

defaultQmod(indices, dfm = 0.6)

defaultN1mod(stock)

defaultVmod(stock, indices)

defaultSRmod(stock)
```

Arguments

`stock`	an FLStock object
`dfm`	numeric vector with the data points fraction to be used to set the spline ks.
`indices`	an FLIndices object

Value

a FLStock object

`deprecated` *deprecated*

Description

Deprecated methods.

Usage

```
a4aSCA(...)
```

Arguments

`...`	additional argument list that might never be used

formula<- *coefficients extract and replacement*

Description

Methods to extract and replace the model coefficients.

Usage

```
formula(object) <- value

## S4 replacement method for signature 'submodel,formula'
formula(object) <- value

coef(object, ...)

## S4 method for signature 'a4aFitSA'
coef(object)

## S4 method for signature 'SCAPars'
coef(object)

## S4 method for signature 'a4aStkParams'
coef(object)

## S4 method for signature 'submodels'
coef(object)

## S4 method for signature 'submodel'
coef(object)

coef(object, ...) <- value

## S4 replacement method for signature 'a4aFitSA,numeric'
coef(object, ...) <- value

## S4 replacement method for signature 'SCAPars,numeric'
coef(object, ...) <- value

## S4 replacement method for signature 'a4aStkParams,numeric'
coef(object, ...) <- value

## S4 replacement method for signature 'submodels,numeric'
coef(object, ...) <- value

## S4 replacement method for signature 'submodel,numeric'
coef(object, ...) <- value
```

```
## S4 replacement method for signature 'submodel,FLPar'
coef(object, ...) <- value

## S4 replacement method for signature 'a4aStkParams,FLPar'
coef(object, ...) <- value

## S4 replacement method for signature 'a4aStkParams,matrix'
coef(object, ...) <- value
```

Arguments

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

---

genFLIndex *Methods to generate FLIndex objects*

Description

This method produces an FLIndex object by using the genFLQuant method.

Usage

```
genFLIndex(object, ...)

## S4 method for signature 'FLQuant'
genFLIndex(object, cv = 0.2, niter = 250)
```

Arguments

object	an FLIndex object
...	additional argument list that might not ever be used.
cv	the coefficient of variation
niter	the number of iterations to be generated

Value

an FLIndex

genFLQuant *Methods to generate FLQuant objects*

Description

This method uses the quant log-correlation matrix of the FLQuant object and generates a new FLQuant using a lognormal multivariate distribution.

Usage

```
genFLQuant(object, ...)

## S4 method for signature 'FLQuant'
genFLQuant(object, cv = 0.2, method = "ac", niter = 250)

## S4 method for signature 'submodel'
genFLQuant(object, type = c("link", "response"), nsim = 0, seed = NULL)

## S4 method for signature 'submodels'
genFLQuant(object, type = c("link", "response"), nsim = 0, seed = NULL)

## S4 method for signature 'a4aStkParams'
genFLQuant(
  object,
  type = c("link", "response"),
  nsim = 0,
  seed = NULL,
  simulate.recruitment = FALSE
)
```

Arguments

object	an FLQuant
...	additional argument list that might not ever be used.
cv	the coefficient of variation
method	the method used to compute the correlation matrix; for now only "ac" (autocorrelation) is implemented
niter	the number of iterations to be generated
type	the type of output required. The default is on the scale of the linear predictors (link); the alternative "response" is on the scale of the response variable. Thus for a model on the log scale the default predictions are of log F (for example) and type = "response" gives the predicted F.
nsim	the number of iterations to simulate, if nsim = 0, then deterministic values are returned based on the coefficients. If nsim > 0 then coefficients are simluated using the covariance slot and distribution slot.

`seed` if supplied the random numbers are generate with a fixed seed for repeatablility

`simulate.recruitment`
if FALSE (default) recruitment is simulated from the recruitment estimates of recruitment, which may or may not be based on a stock-recruit model in the origional fit. If TRUE, then new recruitments are simulated based on the stock recruitment model and supplied CV used in the fit, rsulting in a completly different timeseries of N and Catches.

Value

an FLQuant

Examples

```
data(ple4)
sim.F <- genFLQuant(harvest(ple4))
```

`genFLStock` *Methods to generate FLStock objects*

Description

This method computes the `FLStock` slots consistently with the information provided by the `FLQuant`. It requires two of the triplet R/C/F to compute the third consistent with Baranov and survival's equations.

Usage

```
genFLStock(object, R, C, F, ...)

## S4 method for signature 'FLStock,FLQuant,FLQuant,missing'
genFLStock(object, R, C, F, ...)

## S4 method for signature 'FLStock,missing,FLQuant,FLQuant'
genFLStock(object, R, C, F, ...)

## S4 method for signature 'FLStock,FLQuant,missing,FLQuant'
genFLStock(object, R, C, F, ...)
```

Arguments

`object`	an FLStock
`R`	an FLQuant with iterations or missing
`C`	an FLQuant with iterations or missing
`F`	an FLQuant with iterations or missing
`...`	additional argument list that might not ever be used.

Value

an FLStock

`getAcor` *compute log-correlation matrix*

Description

Method to compute the log-correlation matrix for the first dimension (`quant`) of the `FLQuant` object.

Usage

```
getAcor(object, ...)

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

Arguments

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

Value

an `FLQuant` object with a quant log-correlation matrix

Examples

```
data(ple4)
getAcor(harvest(ple4))
```

`getADMBHessian` *Get ADMB Hessian*

Description

Reads the hessian file from any ADMB fit. Used here with the a4a model.

Usage

```
getADMBHessian(wkdir)

getADMBCovariance(wkdir)
```

Arguments

wkdir	the location of the admb output

Value

a list with the following elements

Note

getADMBHessian is intended to be used internally

Examples

```
# load some data
data(ple4)
data(ple4.indices)
# choose a working directory
wkdir <- tempfile()
# do an 'assessment' fit with default settings (not recomended!) and keep results in wkdir
fit <- sca(stock=ple4,indices=ple4.indices,wkdir=wkdir)
hessInfo <- getADMBHessian(wkdir)
str(hessInfo)
# calculate covariance matrix
Sigma <- solve(hessInfo$hes)
```

---

getCov *Get covariance matrix*

Description

Returns the covariance matrix of the specified Gaussian markov random field model.

Usage

```
getCov(n, model, tau)
```

Arguments

n	integer giving the size of the random feild
model	chatacter giving the name of the GMRF
tau	numeric giving the multiplier of the structure matrix for the model

Value

a covariance matrix

getK *Get K*

Description

Method to get values of the growth parameter K

Usage

```
getK(object, ...)

## S4 method for signature 'a4aGr'
getK(object)
```

Arguments

object	an a4aGr object
...	additional argument list that might never be used

Value

a vector with K values

Examples

```
mm <- matrix(NA, ncol=3, nrow=3)
diag(mm) <- c(50, 0.001,0.001)
mm[upper.tri(mm)] <- mm[lower.tri(mm)] <- c(0.1,0.01,0.00004)
md <- ~linf*(1-exp(-k*(t-t0)))
imd <- ~t0-1/k*log(1-len/linf)
prs <- FLPar(linf=58.5, k=0.086, t0=0.001, units=c("cm","yr^-1","yr"))
vbObj <- a4aGr(grMod=md, grInvMod=imd, params=prs, vcov=mm, distr="norm")
vbObj <- mvrnorm(100,vbObj)
getK(vbObj)
```

getTPL *Get TPL with ADMB code*

Description

Function to get the a4a TPL file with ADMB code and copy into a specific folder.

Usage

```
getTPL(dir)
```

Arguments

dir	folder where the a4a.tpl file will be copied to.

Value

file a4a.tpl

Examples

```
getTPL("myfolder")
```

---

getX *Get model matrix*

Description

Uses the user-specified formula to build a model matrix.

Usage

```
getX(object, ...)

## S4 method for signature 'formula'
getX(object, df, newdf = df)
```

Arguments

object	object of relevant class (see signature of method)
...	additional argument list that might never be used
df	the data.frame to build the model matrix against.
newdf	the data.frame to create the model matrix for.

Value

a matrix.

Note

getX is intended to be used internally

hakeGSA7 *hakeGSA7*

Description

Catch number, stocks weights, etc. for Gulf of Lions Hake (1998 - 2011).

Usage

```
hakeGSA7
```

Format

an FLStock object

Author(s)

Chato Osio

Source

GFCM - STECF

hakeGSA7.idx *hakeGSA7.idx*

Description

Survey index for the Gulf of Lions Hake stock.

Usage

```
hakeGSA7.idx
```

Format

an FLIndices object

Author(s)

Chato Osio

Source

GFCM - STECF - MEDITS

index_cd_len *index_cd_len*

Description

Survey abundance index for hake in the Gulf of Cadiz. Lenth frequencies.

Usage

```
index_cd_len
```

Format

an FLIndex object

Author(s)

Santiago Cervi\~no

index_pt_len *index_pt_len*

Description

Survey abundance index for hake in Portuguese continental waters. Lenth frequencies.

Usage

```
index_pt_len
```

Format

an FLIndex object

Author(s)

Santiago Cervi\~no

index_sp_len *index_sp_len*

Description

Survey abundance index for hake in Northwest Spanish waters. Lenth frequencies.

Usage

```
index_sp_len
```

Format

an FLIndex object

Author(s)

Santiago Cervi\~no

l2a *Method to convert length-based data to age-based*

Description

Method to convert length-based data to age-based

Usage

```
l2a(object, model, ...)

## S4 method for signature 'FLQuant,a4aGr'
l2a(
  object,
  model,
  halfwidth = c(diff(as.numeric(dimnames(object)[[1]])),
    tail(diff(as.numeric(dimnames(object)[[1]])), 1))/2,
  stat = "sum",
  max_age = NA
)

## S4 method for signature 'FLStockLen,a4aGr'
l2a(object, model, plusgroup = NA, ...)

## S4 method for signature 'FLIndex,a4aGr'
l2a(object, model, ...)
```

Arguments

object	an FLQuant, or FLStockLen object.
model	an a4aGr object
...	additional argument list that might never be used
halfwidth	the halfwidths of the length classes; a single numeric or vector the size of the number of length classes; not used if object is an FLStockLen
stat	the aggregation statistic, which must be mean or sum; only used if object is an FLQuant.
max_age	the maximum age in the returned FLQuant; all ages above this are set to max_age; only used if object is an FLQuant
plusgroup	the plusgroup of the stock; only used if the object is an FLStockLen.

Details

A deterministic slicing method converts the length-based data to age-based data, using the supplied growth model (the a4aGr object). Each length-based observation is allocated to a corresponding age, based on the growth model, and aggregated accordingly (either the sum or the mean). There should be 1 or n iterations in both the object being sliced and the growth model. This means that although the slicing method is deterministic, the length-based data is sliced by each iteration of the growth parameters, thereby propagating the uncertainty in the biological growth parameters (representing process uncertainty) through to the age-based data.

Value

an age based FLQuant, FLStock

Examples

```
# Southern hake
# Variance-covariance matrix for parameters
mm <- matrix(NA, ncol=3, nrow=3)
diag(mm) <- c(2310, 0.13, 0.84)
mm[upper.tri(mm)] <- mm[lower.tri(mm)] <- c(-7.22,-6.28,0.08)
# Make the von Bertalanffy growth model
md <- ~linf*(1-exp(-k*(t-t0)))
imd <- ~t0-1/k*log(1-len/linf)
prs <- FLPar(linf=130, k=0.164, t0=-0.092, units=c("cm","yr-1","yr"))
vbObj <- a4aGr(grMod=md, grInvMod=imd, params=prs, vcov=mm, distr="norm")
# Make a triangle copula for simulating process error
linf <- list(a=104.5, b=155.5, c=130)
k <- list(a=0.132, b=0.196, c=0.164)
t0 <- list(a=-0.184, b=0, c=-0.092)
tri_pars <- list(linf = linf, k = k, t0 = t0)
# Simulate 10 iterations from it
vbObj_tri <- mvrtriangle(10, vbObj, paramMargins=tri_pars)
data(southernHakeLen)
# Extract the catch numbers at length from stock object
cth <- catch.n(shake_len)
```

```
# Slice the data using the unsimulated growth object
# so the stock and the growth object have 1 iteration
cthA1 <- l2a(cth, vbObj)
# Slice with 1 iteration in stock and multiple in growth object
cthA2 <- l2a(cth, vbObj_tri)
# Result is age based catch with multiple iterations
# mod: iter=1, data: iter=n
cthA3 <- l2a(propagate(cth,10), vbObj)
# both with iter=n
cthA4 <- l2a(propagate(cth,10), vbObj_tri)
# converting a stock object
shake_age <- l2a(shake_len, vbObj)
shake_age <- l2a(shake_len, vbObj_tri)
shake_age <- l2a(propagate(shake_len, 10), vbObj)
shake_age <- l2a(propagate(shake_len, 10), vbObj_tri)
# converting a index object
index_pt_age <- l2a(index_pt_len, vbObj)
index_pt_age <- l2a(index_pt_len, mvrnorm(10, vbObj))
index_pt_age <- l2a(propagate(index_pt_len, 10), vbObj)
```

---

m *natural mortality*

Description

Method to compute natural mortality.

Usage

```
## S4 method for signature 'a4aM'
m(object, grMod = "missing", ...)
```

Arguments

object	a a4aM object
grMod	a a4aGr object from which the growth parameter K can be extracted
...	placeholder for covariates of the models. The names must match formula variables (not parameters), with the exception of the a4aGr individual growth model. To use a growth model, it must be called grMod and be of class a4aGr, in which case the parameters will be matched. The main objective is to be able to use K from von Bertalanffy models in M.

Details

The method uses the range slot to define the quant and year dimensions of the resulting M FLQuant. The name for the quant dimension is taken as the name of a variable that is present in the shape formula, but not in the params slot of the shape model. If more than one such variable exists, then there is a problem with the shape model definition.

Value

an FLQuant object

Examples

```
age <- 0:15
k <- 0.4
shp <- eval(as.list(~exp(-age-0.5))[[2]], envir=list(age=age))
lvl <- eval(as.list(~1.5*k)[[2]], envir=list(k=k))
M <- shp*lvl/mean(shp)
# Now set up an equivalent a4aM object
mod1 <- FLModelSim(model=~exp(-age-0.5))
mod2 <- FLModelSim(model=~1.5*k, params=FLPar(k=0.4))
m1 <- a4aM(shape=mod1, level=mod2)
  # set up the age range for the object...
  range(m1, c("min", "max")) <- c(0,15)
  # ...and the age range for mbar
  range(m1, c("minmbar", "maxmbar")) <- c(0,15)
m(m1)
mean(m(m1)[ac(0:15)])
all.equal(M, c(m(m1)))

# another example m
range(m1, c("min", "max")) <- c(2,15)
range(m1, c("minmbar", "maxmbar")) <- c(2,4)
m(m1)
mean(m(m1)[ac(2:4)])

# example with specified iters (i.e. not simulated from a statistical distribution)...
mod2 <- FLModelSim(model=~k^0.66*t^0.57,
 params=FLPar(matrix(c(0.4,10,0.5,11), ncol=2, dimnames=list(params=c("k","t"), iter=1:2))),
  vcov=array(c(0.004,0.,0.,0.001,0.006,0.,0.,0.002), dim=c(2,2,2)))
m2 <- a4aM(shape=mod1, level=mod2)
range(m2, c("min", "max")) <- c(2,10)
m(m2)
# ...and with randomly generated iters (based on the medians for params(mod2) and vcov(mod2))
m3 <- a4aM(shape=mod1, level=mvrnorm(100, mod2))
range(m3, c("min", "max")) <- c(0,15)
m(m3)

# example with a trend
mod3 <- FLModelSim(model=~1+b*v, params=FLPar(b=0.05))
mObj <- a4aM(shape=mod1, level=mvrnorm(100, mod2), trend=mod3,
  range=c(min=0,max=15,minyear=2000,maxyear=2003,minmbar=0,maxmbar=0))
m(mObj, v=1:4)
```

---

ma *Model averaging (experimental)*

Description

Method to average across a set of models. This is still experimental. Use with care.

Usage

```
ma(object, ...)

## S4 method for signature 'a4aFitSAs'
ma(object, stock, FUN, nsim = 1000)
```

Arguments

object	an a4aFits object with the fits to be averaged across
...	additional argument list that might never be used
stock	a stock object with the original data used for fitting
FUN	a function to compute the weights, which must return a named vector with weights. Note the weights will be normalized to sum 1 by ma()
nsim	a numeric with the number of simulations to be drawn

Value

an FLStock object with iterations defined by nsim

Examples

```
data(ple4)
data(ple4.indices)
fmod <- ~ factor(age) + s(year, k=20)
qmod <- c(list(~ s(age, k = 4)), rep(list(~s(age, k=4)), 5))
f1 <- sca(ple4, ple4.indices, fmodel=fmod, qmodel=qmod, fit = "assessment")
qmod <- c(list(~ s(age, k = 4) + year), rep(list(~s(age, k=4)), 5))
f2 <- sca(ple4, ple4.indices, fmodel=fmod, qmodel=qmod, fit = "assessment")
# AIC weighting
aicwt <- function(object){
 ICs <- -1 * sapply(object, AIC)
 exp( 0.5 * (ICs - max(ICs)))
}
stock.sim <- ma(a4aFitSAs(list(f1=f1, f2=f2)), ple4, aicwt, nsim = 100)
# equal weighting
eqwt <- function(object){
 v <- rep(1, length(object))
 names(v) <- names(object)
 v
}
stock.sim <- ma(a4aFitSAs(list(f1=f1, f2=f2)), ple4, eqwt, nsim = 100)
```

mvnorm *natural mortality*

Description

Method to simulate multivariate normal parameters for an a4aM object.

Usage

```
## S4 method for signature 'numeric,a4aM,missing,missing,missing,missing'
mvrnorm(n = 1, mu)
```

Arguments

n	the number of iterations to be generated
mu	an a4aM object

Value

an a4aM object with n iterations

Examples

```
mod1 <- FLModelSim(model=~exp(-age-0.5))
mod2 <- FLModelSim(model=~k^0.66*t^0.57, params=FLPar(matrix(c(0.4,10,0.5,11),
 ncol=2, dimnames=list(params=c("k","t"), iter=1:2))),
 vcov=array(c(0.004,0.,0.,0.001,0.006,0.,0.,0.003), dim=c(2,2,2)))
mod3 <- FLModelSim(model=~1+b*v, params=FLPar(b=0.05))
mObj <- a4aM(shape=mod1, level=mod2, trend=mod3,
 range=c(min=0,max=15,minyear=2000,maxyear=2003,minmbar=0,maxmbar=0))
mObj <- mvrnorm(100, mObj)
# Generate 100 iterations with no trend over time
  m(mObj, v=c(1,1,1,1))
# Generate replicates based on iteration-specific multivariate distributions
# (as defined by params() and vcov())
  params(mod2)
  vcov(mod2)
  m1<-mvrnorm(mod2)
  c(params(m1))
# Generate replicates based on a single multivariate distribution (here the
# median of params() and vcov() is used)
  mvrnorm(2,mod2)
  m2<-mvrnorm(2,mod2)
  c(params(m2))
```

mvrcop *Simulation using copula models*

Description

Simulates model parameters with user-defined copulas and marginals.

Usage

```
mvrcop(n, mvdc, ...)

## S4 method for signature 'numeric,FLModelSim'
mvrcop(n, mvdc, copula, ...)
```

Arguments

n	the number of iterations
mvdc	an FLModelSim object
...	arguments to be passed to the copula methods
copula	the name of the copula to be used

Value

an FLModelSim object with n groups of parameters

Examples

```
mm <- matrix(NA, ncol=3, nrow=3)
diag(mm) <- c(100, 0.001,0.001)
mm[upper.tri(mm)] <- mm[lower.tri(mm)] <- c(0.1,0.1,0.0003)
md <- ~linf*(1-exp(-k*(t-t0)))
prs <- FLPar(linf=120, k=0.3, t0=0.1, units=c("cm","yr^-1","yr"))
vb <- FLModelSim(model=md, params=prs, vcov=mm, distr="norm")
pars <- list(list(a=90, b=125, c=120), list(a=0.2, b=0.4), list(a=0, b=0.4, c=0.1))
vbSim <- mvrcop(10000, vb, copula="archmCopula", family="clayton", param=2,
   margins="triangle", paramMargins=pars)
boxplot(t(predict(vbSim, t=0:20+0.5)))
splom(data.frame(t(params(vbSim)@.Data)), pch=".")
```

mvrcop for a4aGr *mvrcop*

Description

Method to generate multivariate parameters with user-defined copulas and marginals for a4aGr objects.

Usage

```
## S4 method for signature 'numeric,a4aGr'
mvrcop(n = 1, mvdc, ...)
```

Arguments

n	the number of iterations
mvdc	the a4aGr object
...	arguments to be passed to the rMvdc and copula methods

Value

an FLModelSim object with n groups of parameters

Examples

```
mm <- matrix(NA, ncol=3, nrow=3)
diag(mm) <- c(50, 0.001,0.001)
mm[upper.tri(mm)] <- mm[lower.tri(mm)] <- c(0.1,0.01,0.00004)
md <- ~linf*(1-exp(-k*(t-t0)))
imd <- ~t0-1/k*log(1-len/linf)
prs <- FLPar(linf=58.5, k=0.086, t0=0.001, units=c("cm","yr^-1","yr"))
vbObj <- a4aGr(grMod=md, grInvMod=imd, params=prs, vcov=mm, distr="norm")
pars <- list(list(a=50, b=100, c=58.5), list(a=0.06, b=0.2, c=0.086), list(a=0, b=0.005, c=0.001))
#In the following, the third, fourth and fifth arguments refer to the copula,
#  while the final two arguments refer to the marginal distributions:
vbObj <- mvrcop(10000, vbObj, copula="archmCopula", family="clayton", param=2,
   margins="triangle", paramMargins=pars)
splom(data.frame(t(params(vbObj)@.Data)), pch=".")
```

mvrnorm for a4aGR *mvrnorm*

Description

Method to generate multivariate normal parameters for a4aGr objects.

Usage

```
## S4 method for signature 'numeric,a4aGr,ANY,ANY,ANY,ANY'
mvrnorm(n = 1, mu)
```

Arguments

n	the number of simulations to be generated
mu	an a4aGr object

Value

an a4aGr object with n iterations

Examples

```
mm <- matrix(NA, ncol=3, nrow=3)
diag(mm) <- c(50, 0.001,0.001)
mm[upper.tri(mm)] <- mm[lower.tri(mm)] <- c(0.1,0.01,0.00004)
md <- ~linf*(1-exp(-k*(t-t0)))
imd <- ~t0-1/k*log(1-len/linf)
prs <- FLPar(linf=58.5, k=0.086, t0=0.001, units=c("cm","yr^-1","yr"))
vbObj <- a4aGr(grMod=md, grInvMod=imd, params=prs, vcov=mm, distr="norm")
vbObj <- mvrnorm(100,vbObj)
```

mvrtriangle *Simulation with a copula model and triangular distributions*

Description

Simulates model parameters using elliptical copulas and triangular marginals.

Usage

```
mvrtriangle(n, object, ...)

## S4 method for signature 'numeric,FLModelSim'
mvrtriangle(n = 1, object, ...)
```

Arguments

n	the number of iterations
object	the FLModelSim object
...	arguments to be passed to the rMvdc and copula methods

Value

an FLModelSim object with n sets of parameters

Examples

```
# Set up the FLModelSim object
mm <- matrix(NA, ncol=3, nrow=3)
diag(mm) <- c(100, 0.001,0.001)
mm[upper.tri(mm)] <- mm[lower.tri(mm)] <- c(0.1,0.1,0.0003)
md <- ~linf*(1-exp(-k*(t-t0)))
prs <- FLPar(linf=120, k=0.3, t0=0.1, units=c("cm","yr^-1","yr"))
vb <- FLModelSim(model=md, params=prs, vcov=mm, distr="norm")
# Simulate from a multivariate normal distribution...
  set.seed(1)
  vbSim <- mvrnorm(10000, vb)
  mm <- predict(vbSim, t=0:20+0.5)
#...from a multivariate triangular distribution with default ranges (0.01 and
#   0.99 quantiles for min and max using a normal distribution with mean from
#   params and sigma from vcov, and with the apex located at params)...
  set.seed(1)
  vbSim1 <- mvrtriangle(10000, vb)
  mm1 <- predict(vbSim1, t=0:20+0.5)
#...and from a multivariate triangular distribution with specified ranges
#   (note if "c" is missing, it will take the average of "a" and "b")
  set.seed(1)
  pars <- list(list(a=90, b=125, c=120), list(a=0.2, b=0.4), list(a=0, b=0.4, c=0.1))
  vbSim2 <- mvrtriangle(10000, vb, paramMargins=pars)
  mm2 <- predict(vbSim2, t=0:20+0.5)
# Plot the results
par(mfrow=c(3,1))
boxplot(t(mm), main="normal")
boxplot(t(mm1), main="triangular")
boxplot(t(mm2), main="triangular2")
splom(data.frame(t(params(vbSim)@.Data)), pch=".")
splom(data.frame(t(params(vbSim1)@.Data)), pch=".")
splom(data.frame(t(params(vbSim2)@.Data)), pch=".")
```

---

mvrtriangle for a4aGr *mvrtriangle*

Description

Method to generate multivariate parameters with elliptical copulas and triangular marginals for a4aGr objects.

Usage

```
## S4 method for signature 'numeric,a4aGr'
mvrtriangle(n = 1, object, ...)
```

Arguments

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

Details

The method is essentially a special case of mvrcop, where the copula is of type "ellipCopula" and family "t", and where the marginals are triangular.

Value

an a4aGr object with n iterations

Examples

```
# Set up the a4aGr object and parameters for the marginals
mm <- matrix(NA, ncol=3, nrow=3)
diag(mm) <- c(50, 0.001,0.001)
mm[upper.tri(mm)] <- mm[lower.tri(mm)] <- c(0.1,0.01,0.00004)
md <- ~linf*(1-exp(-k*(t-t0)))
imd <- ~t0-1/k*log(1-len/linf)
prs <- FLPar(linf=58.5, k=0.086, t0=0.001, units=c("cm","yr^-1","yr"))
vbObj <- a4aGr(grMod=md, grInvMod=imd, params=prs, vcov=mm, distr="norm")
pars <- list(list(a=50, b=100, c=58.5), list(a=0.06, b=0.2, c=0.086), list(a=0, b=0.005, c=0.001))

# Note that mvrtriangle is a special case of mvrcop
set.seed(1)
vbObj1 <- mvrtriangle(10000, vbObj, paramMargins=pars, dispstr="ex", param=0)
set.seed(1)
vbObj2 <- mvrcop(10000, vbObj, copula="ellipCopula", family="t",
 param=0, margins="triangle", paramMargins=pars)
all.equal(vbObj2, vbObj1)
```

---

pars2dim *Check that the second dimension in params is "iter"*

Description

Checks that the name of the second dimension in params is "iter". For internal use, not very interesting for users. It takes a FLModelSim object and returns a logical.

Usage

```
pars2dim(object)

## S4 method for signature 'FLModelSim'
pars2dim(object)

## S4 method for signature 'FLPar'
pars2dim(object)
```

Arguments

object	object of relevant class (see signature of method)

pars2dim

Checks that the name of the second dimension in params is "iter". For internal use and not very interesting for users. It takes an FLPar object and returns a logical.

Examples

```
pars2dim(FLModelSim())
#Example use of pars2dim:
pars2dim(FLPar())
pars2dim(FLPar(array(dim=c(1,1,1))))
```

---

plot for fitted catch-at-age

plot for fitted catch-at-age

Description

Method to plot fitted versus observed catch numbers-at-age. Note the yaxis doesn't has a scale. The visual is about the difference between the two lines, not about the value of each line, which in any case would be very difficult to assess visually.

Usage

```
## S4 method for signature 'a4aFit,FLStock'
plot(x, y, ...)
```

Arguments

x	an a4aFit object with the fitted values
y	an FLStock object with the observed values
...	additional argument list that might never be used

Value

a `plot` with fitted and observed catch numbers-at-age

Examples

```
data(ple4)
data(ple4.index)
obj <- sca(ple4, FLIndices(ple4.index))
plot(obj, ple4)
```

`plot for fitted indices-at-age`
testing

Description

Method to plot fitted versus observed indices-at-age. Note the yaxis doesn't has a scale. The visual is about the difference between the two lines, not about the value of each line, which in any case would be very difficult to assess visually.

Usage

```
## S4 method for signature 'a4aFit,FLIndices'
plot(x, y, ...)
```

Arguments

`x`	an `a4aFit` object with the fitted values
`y`	an `FLIndices` object with the observed values
`...`	additional argument list that might never be used

Value

a `plot` with fitted and observed indices-at-age

Examples

```
data(ple4)
data(ple4.index)
obj <- sca(ple4, FLIndices(ple4.index))
plot(obj, FLIndices(ple4.index))
```

`plot of residuals` *Plot of standardized log residuals*

Description

Method to produce scatterplots of standardized residuals

Method to produce scatterplots of standardized residuals

Usage

```
## S4 method for signature 'a4aFitResiduals,missing'
plot(x, y = missing, auxline = "smooth", ...)

## S4 method for signature 'a4aFitCatchDiagn,missing'
plot(x, y = missing, ...)
```

Arguments

`x`	an `a4aFitResiduals` object with the standardized residuals
`y`	ignored
`auxline`	a string defining the type of line to be added, by default uses 'smooth', a common alternative is to use 'r', a regression, or leave it empty "
`...`	additional argument list that might never be used

Value

a `plot` with stardardized log residuals

a `plot` with stardardized log residuals

Examples

```
data(ple4)
data(ple4.index)
obj <- sca(ple4, FLIndices(ple4.index))
flqs <- residuals(obj, ple4, FLIndices(idx=ple4.index))
plot(flqs)
data(ple4)
data(ple4.index)
obj <- sca(ple4, FLIndices(ple4.index))
flqs <- residuals(obj, ple4, FLIndices(idx=ple4.index))
plot(flqs)
```

predict for a4aGr *predict for* a4aGr

Description

Predicts ages or lengths using a growth class

Usage

```
## S4 method for signature 'a4aGr'
predict(object, ...)
```

Arguments

object	the a4aGr object
...	arguments to be passed to the rMvdc and copula methods

Value

a matrix object with lengths or ages

Examples

```
# Set up the a4aGr object and parameters for the marginals
mm <- matrix(NA, ncol=3, nrow=3)
diag(mm) <- c(50, 0.001,0.001)
mm[upper.tri(mm)] <- mm[lower.tri(mm)] <- c(0.1,0.01,0.00004)
md <- ~linf*(1-exp(-k*(t-t0)))
imd <- ~t0-1/k*log(1-len/linf)
prs <- FLPar(linf=58.5, k=0.086, t0=0.001, units=c("cm","yr^-1","yr"))
vbObj <- a4aGr(grMod=md, grInvMod=imd, params=prs, vcov=mm, distr="norm")
predict(vbObj, len=1:50+0.5)
predict(vbObj, t=1:20+0.5)
```

predict for sca *Predict methods for SCA*

Description

Predict methods for a4a stock assessment fits.

Usage

```
## S4 method for signature 'a4aFitSA'
predict(object)

## S4 method for signature 'SCAPars'
predict(object)
```

Arguments

object	object of relevant class (see signature of method)

Examples

```
data(ple4)
data(ple4.index)
fmodel <- ~factor(age) + factor(year)
qmodel <- list(~factor(age))
fit1 <-  sca(fmodel=fmodel, qmodel=qmodel, stock=ple4, indices=FLIndices(ple4.index))
flqs <- predict(fit1)
```

---

qqplot of residuals *qqplot of standardized log residuals*

Description

Method to produce qqplots of standardized residuals

Usage

```
## S4 method for signature 'a4aFitResiduals,missing'
qqmath(x, data = missing, ...)
```

Arguments

x	an a4aFitResiduals object with the standardized residuals
data	ignored
...	additional argument list that might never be used

Value

a qqplot with stardardized log residuals

Examples

```
data(ple4)
data(ple4.index)
obj <- sca(ple4, FLIndices(ple4.index))
flqs <- residuals(obj, ple4, FLIndices(idx=ple4.index))
qqmath(flqs)
```

range<-,a4aM,ANY,numeric-method *range for a4aM objects*

Description

Range method for a4aM objects

Usage

```
## S4 replacement method for signature 'a4aM,ANY,numeric'
range(x, i) <- value
```

Arguments

x	an a4aM object
i	the elements of range to be changed in a character vector
value	a numeric vector with values

rfLen *redfish length data*

Description

Simulated length data for redfish. Simulations were done using GADGET.

Usage

```
data(rfLen)
```

Format

An FLStock.

Author(s)

Ernesto Jardim

Source

Daniel Howell

rfLen.stk *rfLen.stk*

Description

Simulated stock based on red fish.

Usage

```
rfLen.stk
```

Format

an FLStock object

Author(s)

Daniel Howell

rfTrawl.idx *rfTrawl.idx*

Description

Trawl survey index for red fish.

Usage

```
rfTrawl.idx
```

Format

an FLIndex object

Author(s)

Daniel Howell

rfTrawlJmp.idx *rfTrawlJmp.idx*

Description

Trawl survey index for red fish, with a jump in catchability.

Usage

```
rfTrawlJmp.idx
```

Format

an FLIndex object

Author(s)

Daniel Howell

rfTrawlTrd.idx *rfTrawlTrd.idx*

Description

Trawl survey index for red fish, with a trend in catchability.

Usage

```
rfTrawlTrd.idx
```

Format

an FLIndex object

Author(s)

Daniel Howell

sca *Statistical catch-at-age method*

Description

Statistical catch-at-age method of the a4a stock assessment framework.

Usage

```
sca(stock, indices, ...)

## S4 method for signature 'FLStock,FLIndex'
sca(stock, indices, ...)

## S4 method for signature 'FLStock,FLIndices'
sca(
  stock,
  indices,
  fmodel = missing,
  qmodel = missing,
  srmodel = missing,
  n1model = missing,
  vmodel = missing,
  covar = missing,
  wkdir = missing,
  verbose = FALSE,
  fit = "assessment",
  center = TRUE,
  mcmc = missing
)
```

Arguments

stock	an FLStock object containing catch and stock information
indices	an FLIndices object containing survey indices
...	additional argument list that might never be used
fmodel	a formula object depicting the model for log fishing mortality at age
qmodel	a list of formula objects depicting the models for log survey catchability at age
srmodel	a formula object depicting the model for log recruitment
n1model	a formula object depicting the model for the population in the first year of the time series
vmodel	a list of formula objects depicting the model for the variance of fishing mortality and the indices
covar	a list with covariates to be used by the submodels. The formula must have an element with the same name as the list element.

wkdir	used to set a working directory for the admb optimiser; if wkdir is set, all admb files are saved to this folder, otherwise they are deleted.
verbose	if true, admb fitting information is printed to the screen.
fit	character with type of fit: 'MP' or 'assessment'; the former does not require the hessian to be computed, while the latter does.
center	logical defining if the data should be centered before fitting.
mcmc	an SCAMCMC object with the arguments to run MCMC

Details

[REQUIRES REVISION] This method is the advanced method for stock assessment, it gives the user access to a set of arguments that the sca method doesn't. In particular, the default for the fit argument is 'assessment'. For detailed information about using the sca read the vignette 'The a4a Stock Assessment Modelling Framework' (vignette('sca')).

Value

an a4aFit object if fit is "MP" or an a4aFitSA object if fit is "assessment"

Examples

```
data(ple4)
data(ple4.index)

# fishing mortality by age and year (separable) AND catchability at age without year trend
fmodel <- ~factor(age) + factor(year)
qmodel <- list(~factor(age))
fit1 <-  sca(fmodel=fmodel, qmodel=qmodel, stock=ple4, indices=FLIndices(ple4.index))

# fishing mortality as a smoother by age and year (but still separable) AND
# catchability at age without year trend
fmodel <- ~ s(age, k=4) + s(year, k=10)
qmodel <- list(~factor(age))
fit2 <-  sca(fmodel=fmodel, qmodel=qmodel, stock=ple4, indices=FLIndices(ple4.index))

# fishing mortality as a smoother by age and year (but still separable) AND
# catchability as a smoother by age without year trend
fmodel <- ~ s(age, k=4) + s(year, k=10)
qmodel <- list(~s(age, k=4))
fit3 <-  sca(fmodel=fmodel, qmodel=qmodel, stock=ple4, indices=FLIndices(ple4.index))

# fishing mortality as a smoother by age and year (but still separable) AND
# catchability as a smoother by age with year trend
fmodel <- ~ s(age, k=4) + s(year, k=10)
qmodel <- list(~s(age, k=4) + year)
fit4 <-  sca(fmodel=fmodel, qmodel=qmodel, stock=ple4, indices=FLIndices(ple4.index))

# It's a statistical model
BIC(fit1, fit2, fit3, fit4)
```

```
# fishing mortality as a smoother by age and year with interactions (i.e. non-separable) AND
# catchability as a smoother by age without year trend
fmodel <- ~ te(age, year, k=c(4, 10))
qmodel <- list(~s(age, k=4))
fit5 <-  sca(fmodel=fmodel, qmodel=qmodel, stock=ple4, indices=FLIndices(ple4.index))

# fit3 + smoother in recruitment
fmodel <- ~ s(age, k=4) + s(year, k=20)
qmodel <- list(~s(age, k=4))
rmodel <- ~s(year, k=20)
fit6 <-  sca(fmodel=fmodel, qmodel=qmodel, srmodel=rmodel, ple4, FLIndices(ple4.index))

# fit3 + bevholt
rmodel <- ~ bevholt(CV=0.05)
fit7 <-  sca(fmodel=fmodel, qmodel=qmodel, srmodel=rmodel, ple4, FLIndices(ple4.index))
```

`sca.sa` *Call sca inside the mp function*

Description

This function provides an interface to sca() to be used inside the mp() function of the mse package.

Usage

```
sca.sa(stk, idx, update = TRUE, dfm = c(0.75, 0.75), args, tracking, ...)
```

Arguments

`stk`	The FLStock input object.
`idx`	The FLIndices input object.
`update`	Should the fmodel be updated with the default?
`dfm`	data points fraction to be used to set the spline ks.
`...`	Any other arguments to sca()
`genArgs`	The mse arguments used by mp().

Value

A list containing the estimated stock (stk, of class FLStock), and the tracking FLQuant, including convergence flags.

SCAMCMC *MCMC settings class*

Description

Class definition (slots), constructors, accessors, replacement (when relevant) and common methods.

Usage

```
SCAMCMC(object, ...)

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

getADMBCallArgs(object, ...)

## S4 method for signature 'SCAMCMC'
getADMBCallArgs(object, ...)

getN(object, ...)

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

Arguments

object	a SCAMCMC object
...	extra arguments

Slot

mcmc N Run N MCMC iterations

mcsave N Save every N th MCMC iteration

mcscale N Rescale step size for first N iterations

mcmult N Rescale the covariance matrix

mcrb N Reduce high parameter correlations

mcprobe X Use a fat-tailed proposal distribution

mcdiag Use a diagonal covariance matrix

mcnoscale Do not scale the algorithm during

mcu Use a uniform distribution as proposal distribution

hybrid Use the hybrid method

hynstep N Mean number of steps for the leapfrog method

hyeps X The stepsize for the leapfrog method [X numeric and > 0]

Accessors

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

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

Constructor

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

SCAPars *Model parameters class*

Description

Class definition (slots), constructors, accessors, replacement (when relevant) and common methods.

Usage

```
SCAPars(object, ...)

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

stkmodel(object, ...)

## S4 method for signature 'SCAPars'
stkmodel(object)

n1model(object, ...)

## S4 method for signature 'SCAPars'
n1model(object)

srmodel(object, ...)

## S4 method for signature 'SCAPars'
srmodel(object)

fmodel(object, ...)

## S4 method for signature 'SCAPars'
fmodel(object)
```

```
qmodel(object, ...)

## S4 method for signature 'SCAPars'
qmodel(object)

qMod(object, ...)

## S4 method for signature 'SCAPars'
qMod(object)

vmodel(object, ...)

## S4 method for signature 'SCAPars'
vmodel(object)

vMod(object, ...)

## S4 method for signature 'SCAPars'
vMod(object)

srPars(object, ...)

## S4 method for signature 'SCAPars'
srPars(object)

srCovar(object, ...)

## S4 method for signature 'SCAPars'
srCovar(object)

srFrml(object, ...)

## S4 method for signature 'SCAPars'
srFrml(object)

fPars(object, ...)

## S4 method for signature 'SCAPars'
fPars(object)

fCovar(object, ...)

## S4 method for signature 'SCAPars'
fCovar(object)

fFrml(object, ...)
```

```
## S4 method for signature 'SCAPars'
fFrml(object)

qPars(object, ...)

## S4 method for signature 'SCAPars'
qPars(object)

qCovar(object, ...)

## S4 method for signature 'SCAPars'
qCovar(object)

qFrml(object, ...)

## S4 method for signature 'SCAPars'
qFrml(object)

vPars(object, ...)

## S4 method for signature 'SCAPars'
vPars(object)

vCovar(object, ...)

## S4 method for signature 'SCAPars'
vCovar(object)

vFrml(object, ...)

## S4 method for signature 'SCAPars'
vFrml(object)

## S4 method for signature 'SCAPars'
m(object)

## S4 method for signature 'SCAPars'
wt(object)

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

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

Arguments

object	object of relevant class (see signature of method)

...	additional argument list that might never be used
iter	the number of iterations to create
fill.iter	should the new iterations be filled with values (TRUE) or NAs (FALSE)
obj	the object to be subset
it	iteration to be extracted

Slot

stkmodel parameters related to stock dynamics

qmodel paramaters related to catchability of tunning fleets

vmodel paramaters related to the variance model

Accessors

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

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

Constructor

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

---

sep.sa *Call a separable SA inside the mp function*

Description

This function provides an interface to a call to a separable model based on sca() to be used inside the mp() function of the mse package.

Usage

```
sep.sa(stk, idx, args, update = TRUE, dfm = c(0.75, 0.75), ...)
```

Arguments

stk	The FLStock input object.
idx	The FLIndices input object.
update	Should the fmodel be updated with the default?
dfm	data points fraction to be used to set the spline ks.
...	Any other arguments to sca()
genArgs	The mse arguments used by mp().

Value

A list containing the estimated stock (stk, of class FLStock), and the tracking FLQuant, including convergence flags.

`shake_len` *shake_len*

Description

FLR stock object for southern hake.

Usage

```
shake_len
```

Format

an `FLStock` object

Author(s)

Santiago Cervi\~no

`simulate` *Simulation methods for SCA*

Description

Simulation methods for a4a stock assessment fits.

Usage

```
simulate(object, nsim = 1, seed = NULL, ...)

## S4 method for signature 'a4aFitSA'
simulate(object, nsim = 1, seed = NULL, empirical = TRUE)

## S4 method for signature 'SCAPars'
simulate(object, nsim = 1, seed = NULL, empirical = TRUE)

## S4 method for signature 'a4aStkParams'
simulate(object, nsim = 1, seed = NULL, empirical = TRUE)

## S4 method for signature 'submodels'
simulate(object, nsim = 1, seed = NULL, empirical = TRUE)

## S4 method for signature 'submodel'
simulate(object, nsim = 1, seed = NULL, empirical = TRUE)
```

Arguments

object	object of relevant class (see signature of method)
nsim	number of iterations
seed	numeric with random number seed
...	additional argument list that might never be used
empirical	logical, shall the empirical method in MASS be used

Examples

```
data(ple4)
data(ple4.index)
fmodel <- ~factor(age) + factor(year)
qmodel <- list(~factor(age))
fit1 <-  sca(fmodel=fmodel, qmodel=qmodel, stock=ple4, indices=FLIndices(ple4.index))
fit1
summary(fit1)
stock.n(fit1)
```

---

southernHakeLen *Southern hake length data*

Description

Length based stock and three indices data for Southern hake.

Usage

```
data(southernHakeLen)
```

Format

an FLStockLen and three FLIndex objects.

Author(s)

Finlay Scott

Source

Santiago Cervino

stdlogres *Standardized log residuals*

Description

Method to compute the standardized residuals on the log scale for index- and catch-at-age residuals in the a4a stock assessment framework.

Usage

```
stdlogres(obs, fit, ...)

## S4 method for signature 'FLQuant,FLQuant'
stdlogres(obs, fit, ...)
```

Arguments

obs	an FLQuant object with the observations
fit	an FLQuant object with the fitted value
...	additional argument list that might never be used

Value

an FLQuant with stardardized log residuals

Examples

```
data(ple4)
data(ple4.index)
obj <- sca(ple4, FLIndices(ple4.index))
flqs <- residuals(obj, ple4, FLIndices(idx=ple4.index))
stdlogres(catch.n(ple4), catch.n(obj))
# which is the same as the following (because residuals() uses stdlogres):
flqs$catch.n
# check:
stdlogres(catch.n(ple4),catch.n(obj)) - flqs$catch.n
```

submodel *Submodel class*

Description

Class definition (slots), constructors, accessors, replacement (when relevant) and common methods.

Usage

```
submodel(object, ...)

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

## S4 method for signature 'submodel'
params(object)

sMod(object, ...)

## S4 method for signature 'submodel'
sMod(object)

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

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

## S4 method for signature 'submodel'
formula(x)
```

Arguments

object	object of relevant class (see signature of method)
...	additional argument list that might never be used
obj	the object to be subset
it	iteration to be extracted
iter	the number of iterations to create
fill.iter	should the new iterations be filled with values (TRUE) or NAs (FALSE)
x	the submodel object that is to be modified

Slot

`Mod` `formula` describing the model

`params` `FLPar` with model parameters

`vcov` `array` with variance covariance paramaters related to the variance model

`centering` `numeric` value used for centering the data

`distr` a character with the parameters' statistical distribution; it must match a known distribution for R (*e.g.* "norm" for gaussian) so that `rnorm` can be called

Accessors

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

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

Constructor

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

`submodels` *Submodels class*

Description

Class definition (slots), constructors, accessors, replacement (when relevant) and common methods.

Usage

```
submodels(...)

submodels(...)

corBlocks(object, ...)

## S4 method for signature 'submodels'
corBlocks(object)

## S4 method for signature 'submodels'
params(object)

## S4 method for signature 'submodels'
sMod(object)

## S4 method for signature 'submodels'
formula(x)

corBlocks(object, ...) <- value

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

## S4 replacement method for signature 'submodels,submodel'
```

```
x$name <- value

## S4 replacement method for signature 'submodels,character,missing'
x[[i, j, ...]] <- value

## S4 replacement method for signature 'submodels,numeric,missing'
x[[i, j, ...]] <- value

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

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

Arguments

...	additional argument list that might never be used
object	object of relevant class (see signature of method)
x	object to be modified
value	value the new object
name	name(s) of entry to be extracted / modified
i, j	indices specifying elements to extract or replace.
iter	the number of iterations to create
fill.iter	should the new iterations be filled with values (TRUE) or NAs (FALSE)
obj	the object to be subset
it	iteration to be extracted

Constructor

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

Note

This class is similar to other 'plural' calsses in FLR. It is a list constrained to having all elements of the same class, in this case submodel. Otherwise it works exacly as any other list.

vcov *Variance-covariance matrix*

Description

Methods to extract and replace the variance-covariance matrix.

Usage

```
## S4 method for signature 'a4aFitSA'
vcov(object)

## S4 method for signature 'SCAPars'
vcov(object)

## S4 method for signature 'submodels'
vcov(object)

## S4 method for signature 'submodel'
vcov(object)

## S4 replacement method for signature 'a4aFitSA,numeric'
vcov(object, ...) <- value

## S4 replacement method for signature 'SCAPars,numeric'
vcov(object, ...) <- value

## S4 replacement method for signature 'a4aStkParams,numeric'
vcov(object, ...) <- value

## S4 replacement method for signature 'submodel,numeric'
vcov(object, ...) <- value

## S4 replacement method for signature 'submodel,matrix'
vcov(object, ...) <- value

## S4 replacement method for signature 'submodel,array'
vcov(object, ...) <- value
```

Arguments

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

wireframe plot for FLQuant
wireframe plot for FLQuant

Description

Method to 3D plot FLQuant objects.

Usage

```
## S4 method for signature 'FLQuant,missing'
wireframe(x, y, screen = list(x = -90, y = -45), ...)
```

Arguments

x	a FLQuant
y	missing
screen	list with numeric components 'x','y' and 'z' to change the 3D perspective
...	additional argument list for the lattice engine

Value

a 3D surface plot

Examples

```
data(ple4)
wireframe(harvest(ple4))
```

Index