Package: stockassessment (via r-universe)

October 23, 2024

Title State-Space Assessment Model

Version 0.12.4

Date 2023-10-09

Author Anders Nielsen [aut, cre], Casper Berg [aut], Christoffer Moesgaard Albertsen [aut], Kasper Kristensen [aut], Mollie Brooks [aut], Vanessa Trijoulet [aut], Olav Nikolai Breivik [aut]

Maintainer Anders Nielsen <an@aqua.dtu.dk>

Description Fitting SAM...

License GPL-2

Depends R (>= 3.0.2)

Imports TMB, ellipse, methods, stats, utils, grDevices, graphics, Matrix

LinkingTo TMB (>= 1.9.1), RcppEigen

Encoding UTF-8

URL https://github.com/fishfollower/SAM

LazyData TRUE

BugReports https://github.com/fishfollower/SAM/issues

SystemRequirements GNU make

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

RemoteUrl https://github.com/fishfollower/SAM

RemoteRef rp_rebase

RemoteSha 1e60e3ca566eaa1208f036b68cdcde9ceb5df712

Contents

.SAM_replicate 4
addforecast 5
addRecruitmentCurve 6

b0plot . . . 7
b0table . . . 8
bc . . . 9
c.sam . . . 9
catchbyfleetplot . . . 10
catchbyfleettable . . . 10
catchplot . . . 11
catchtable . . . 12
caytable . . . 12
clean.void.catches . . . 13
coef.sam . . . 13
componentplot . . . 14
corplot . . . 15
corplotcommon . . . 15
dataplot . . . 16
defcon . . . 16
defpar . . . 17
deterministicReferencepoints . . . 18
empirobscorrplot . . . 19
equilibriumbiomassplot . . . 20
equilibriumbiomasstable . . . 21
erbplot . . . 21
erbtable . . . 22
faytable . . . 22
fbarplot . . . 23
fbartable . . . 25
fitfromweb . . . 25
fitplot . . . 26
forecast . . . 26
forecastMSY . . . 28
fselectivityplot . . . 30
generationlengthplot . . . 30
generationlengthtable . . . 31
getAllDerivedValues . . . 31
getFleet . . . 32
getLowerBounds . . . 32
getResidualFleets . . . 33
getUpperBounds . . . 33
grad . . . 34
hcr . . . 34
hessian . . . 36
ibc . . . 37
icesAdviceRule . . . 37
iibc . . . 38
is.whole.positive.number . . . 39
jacobian . . . 40
jit . . . 40
leaveout . . . 41

lifeexpectancyplot 42
lifeexpectancytable 43
loadConf 43
logLik.sam 44
modelDescription 44
modelforecast 45
modeltable 51
modelVersionInfo 51
mohn 52
MSE 52
nobs.sam 53
nscodConf 54
nscodData 55
nscodParameters 55
ntable 56
obscorrplot 57
obscov 57
parplot 58
partable 58
plot.hcr 59
plot.sam 59
plot.samforecast 60
plot.samres 60
plot.samset 61
plot.samypr 61
plotby 62
plotit 63
predstdplot 66
print.hcr 67
print.sam 67
print.samcoef 68
print.samforecast 68
print.samres 69
print.samset 69
print.samypr 70
print.sam_referencepoints 70
procres 71
qtable 71
qtable.sam 72
qtableplot 72
read.data.files 73
read.ices 73
read.surveys 74
read.table.nowarn 74
recplot 75
rectable 75
reduce 76
referencepoints 77

refit 79
residuals.sam 79
retro 80
rmaxplot 80
rmaxtable 81
rmvnorm 82
runwithout 82
sam.fit 83
saveConf 87
sdplot 88
setS 89
setSeq 89
setup.sam.data 90
simstudy 91
simulate.sam 92
sprplot 93
sprtable 93
srplot 94
ssbplot 95
ssbtable 96
stochasticReferencepoints 96
stockassessment-deprecated 99
summary.sam 99
tableit 100
tsbplot 100
tsbtable 101
write.data.files 101
write.ices 102
write.surveys 103
yearslostplot 103
yearslosttable 104
ypr 105
yprplot 106
yprtable 106

Index **108**

`.SAM_replicate` *Parallel replicate for modelforecast*

Description

Parallel replicate for modelforecast

Usage

```
.SAM_replicate(
  n,
  expr,
  simplify = "array",
  ncores = 1,
  env = parent.frame(n + 1),
  par_precall = NULL,
  type = ifelse(.Platform$OS.type == "unix", "mclapply", "PSOCK")
)
```

Arguments

`n`	number of replicates
`expr`	expression
`simplify`	simplify passes to sapply
`ncores`	number of cores
`env`	environment
`par_precall`	Code to run when starting
`type`	type of parallelisation

Value

output

`addforecast` *SAM add forecasts*

Description

SAM add forecasts

Usage

```
addforecast(
  fit,
  what,
  dotcol = "black",
  dotpch = 19,
  dotcex = 1.5,
  intervalcol = gray(0.5, alpha = 0.5),
  ...
)

## S3 method for class 'samforecast'
```

```
addforecast(
  fit,
  what,
  dotcol = "black",
  dotpch = 19,
  dotcex = 1.5,
  intervalcol = gray(0.5, alpha = 0.5),
  ...
)
```

Arguments

`fit`	the object returned from sam.fit
`what`	what to plot
`dotcol`	color for dot
`dotpch`	pch for dot
`dotcex`	cex for dot
`intervalcol`	color for interval
`...`	extra arguments not currently used

Details

internal plotting fun

`addRecruitmentCurve` *Add stock-recruitment curve to srplot*

Description

Add stock-recruitment curve to srplot

Usage

```
addRecruitmentCurve(
  fit,
  CI = TRUE,
  col = rgb(0.6, 0, 0),
  cicol = rgb(0.6, 0, 0, 0.3),
  plot = TRUE,
  PI = FALSE,
  picol = rgb(0.6, 0, 0),
  pilty = 2,
  ...
)
```

```
## S3 method for class 'sam'
addRecruitmentCurve(
  fit,
  CI = TRUE,
  col = rgb(0.6, 0, 0),
  cicol = rgb(0.6, 0, 0, 0.3),
  plot = TRUE,
  PI = FALSE,
  picol = rgb(0.6, 0, 0),
  pilty = 2,
  year = NA_real_,
  lastR = NA_real_,
  ...
)
```

Arguments

fit	Object to show SR-curve for
CI	Add confidence intervals?
col	Color of fitted line
cicol	Color of confidence intervals
plot	Add the curve to a plot?
PI	Add prediction intervals?
picol	Color of prediction interval line
pilty	Line type of prediction interval line
...	not used
year	Show recruitment calculated conditional on this year (for recruitment functions that depend on year)
lastR	Show recruitment calculated conditional on this previous recruitment (for recruitment functions that depend on recruitment the previous year)

See Also

srplot

b0plot *SAM equilibrium biomass in the absence of fishing plot*

Description

SAM equilibrium biomass in the absence of fishing plot

Usage

```
b0plot(fit, ...)

## Default S3 method:
b0plot(fit, ...)

## S3 method for class 'samforecast'
b0plot(fit, ...)

## S3 method for class 'hcr'
b0plot(fit, ...)
```

Arguments

`fit` the object returned from sam.fit

`...` extra arguments transferred to plot including the following:
`add` logical, plotting is to be added on existing plot
`ci` logical, confidence intervals should be plotted
`cicol` color to plot the confidence polygon

Details

Plot of deterministic equilibrium biomass in the absence of fishing assuming biological parameters and selectivity for that year remains unchanged in the future.

`b0table` *B0 biomass table*

Description

B0 biomass table

Usage

```
b0table(fit, ...)

## Default S3 method:
b0table(fit, ...)
```

Arguments

`fit` ...

`...` extra arguments not currently used

Details

...

`bc` *Spline basis for use with formula interface*

Description

Spline basis for use with formula interface

Usage

```
bc(x, df = 3L, knots = NULL, Boundary.knots = range(x), intercept = FALSE)
```

Arguments

`x`	Points to evaluate the basis in
`df`	Degrees of freedom
`knots`	Internal knots. If NULL, they are selected from quantiles of x.
`Boundary.knots`	Boundary knots. Defaults to range of x
`intercept`	Include an intercept in basis?

Value

A spline basis

`c.sam` *Collect sam objects*

Description

Collect sam objects

Usage

```
## S3 method for class 'sam'
c(...)
```

Arguments

`...`	one or more sam fits (as returned from the `sam.fit` function) to be combined

Details

...

`catchbyfleetplot` *SAM catchbyfleet plot*

Description

SAM catchbyfleet plot

Usage

```
catchbyfleetplot(fit, obs.show = FALSE, ...)
```

Arguments

`fit`	the object returned from sam.fit
`obs.show`	if observations are to be shown also
`...`	extra arguments transferred to plot

Details

Plot of estimated (and optionally observed) total catch in weight

`catchbyfleettable` *CatchByFleet table*

Description

CatchByFleet table

Usage

```
catchbyfleettable(fit, obs.show = FALSE)
```

Arguments

`fit`	object returned from sam.fit
`obs.show`	logical add a column with catch sum of product rowsums(C*W)

Details

...

catchplot *SAM catch plot*

Description

SAM catch plot

Usage

```
catchplot(fit, obs.show = TRUE, drop = NULL, ...)

## S3 method for class 'sam'
catchplot(fit, obs.show = TRUE, drop = NULL, ...)

## S3 method for class 'samset'
catchplot(fit, obs.show = TRUE, drop = NULL, ...)

## S3 method for class 'samforecast'
catchplot(fit, obs.show = TRUE, drop = NULL, ...)

## S3 method for class 'hcr'
catchplot(fit, obs.show = TRUE, drop = NULL, ...)
```

Arguments

`fit`	the object returned from sam.fit
`obs.show`	if observations are to be shown also
`drop`	number of years to be left unplotted at the end. Default (NULL) is to not show years at the end with no catch information
`...`	extra arguments transferred to plot including the following: `add` logical, plotting is to be added on existing plot `ci` logical, confidence intervals should be plotted `cicol` color to plot the confidence polygon

Details

Plot of estimated (and optionally observed) total catch in weight

`catchtable` *Catch table*

Description

Catch table

Usage

```
catchtable(fit, obs.show = FALSE, ...)

## S3 method for class 'sam'
catchtable(fit, obs.show = FALSE, ...)
```

Arguments

`fit`	object returned from sam.fit
`obs.show`	logical add a column with catch sum of product rowsums(C*W)
`...`	extra arguments not currently used

Details

...

`caytable` *Catch-at-age in numbers table*

Description

Catch-at-age in numbers table

Usage

```
caytable(fit, fleet = which(fit$data$fleetTypes == 0))
```

Arguments

`fit`	a fitted object of class ’sam’ as returned from sam.fit
`fleet`	the fleet number(s) to return catch summed for (default is to return the sum of all residual fleets).

Details

...

`clean.void.catches` *remove void catches*

Description

remove void catches

Usage

```
clean.void.catches(dat, conf)
```

Arguments

`dat`	data for the sam model as returned from the setup.sam.data function
`conf`	model configuration which can be set up using the `defcon` function and then modified

Value

an updated dataset without the catches where F is fixed to zero

`coef.sam` *Extract fixed coefficients of sam object*

Description

Extract fixed coefficients of sam object

Usage

```
## S3 method for class 'sam'
coef(object, ...)
```

Arguments

`object`	sam fitted object as returned from the `sam.fit` function
`...`	extra arguments

Details

fixed coefficients of sam object

componentplot *Area plot of spawning components*

Description

Area plot of spawning components

Usage

```
componentplot(fit, ...)

## S3 method for class 'sam'
componentplot(
  fit,
  onlyComponentYears = FALSE,
  ylab = "Composition",
 colSet = c("#332288", "#88CCEE", "#44AA99", "#117733", "#999933", "#DDCC77", "#661100",
    "#CC6677", "#882255", "#AA4499"),
  legend.pos = "bottom",
  bg = "white",
  ncol = length(cf),
  ...
)
```

Arguments

`fit`	sam fit
`...`	passed to legend
`onlyComponentYears`	If true, x axis is limited to the range with spawning component data. Otherwise, the model years are used.
`ylab`	Label for y axis
`colSet`	Colors
`legend.pos`	Legend position. See ?legend
`bg`	Background of legend. See ?legend
`ncol`	Number of columns in legend. See ?legend

Value

Nothing

Author(s)

Christoffer Moesgaard Albertsen

corplot — *Plots between-age correlations by fleet, either estimated or empirical using residuals.*

Description

Plots between-age correlations by fleet, either estimated or empirical using residuals.

Usage

```
corplot(x, ...)

## S3 method for class 'sam'
corplot(x, ...)

## S3 method for class 'samres'
corplot(x, ...)
```

Arguments

`x`	Either a sam fit as returned by sam.fit OR the object returned from residuals.sam
`...`	extra arguments to plot

corplotcommon — *Common function for plotting correlation matrices.*

Description

Common function for plotting correlation matrices.

Usage

```
corplotcommon(x, fn, ...)
```

Arguments

`x`	a list of correlation matrices
`fn`	a vector of fleet names
`...`	extra arguments to plotcorr

dataplot *SAM Data plot*

Description

SAM Data plot

Usage

```
dataplot(fit, col = NULL, fleet_type = NULL, fleet_names = NULL)

## S3 method for class 'sam'
dataplot(fit, col = NULL, fleet_type = NULL, fleet_names = NULL)
```

Arguments

`fit`	the object returned from sam.fit
`col`	color to use for each fleet, default is two sequential colors
`fleet_type`	character vector giving the type of data per fleet. The default uses fit$data$fleetTypes as follows: `fit$data$fleetTypes==0` "Catch at age" `fit$data$fleetTypes==1` "Catch at age with effort" `fit$data$fleetTypes==2` or `6` "Index at age" `fit$data$fleetTypes==3` "Biomass or catch index" `fit$data$fleetTypes==5` "Tagging data" `fit$data$fleetTypes==7` "Sum of fleets"
`fleet_names`	character vector giving fleet names. The default is given by attr(fit$data,"fleetNames")

Details

Plot data available for the stock

defcon *Setup basic minimal configuration for sam assessment*

Description

Setup basic minimal configuration for sam assessment

Usage

```
defcon(dat, level = 1)
```

Arguments

`dat`	sam data object
`level`	1 or 2 (1 most basic configuration, 2 configuration with AR correlation structure on surveys)

Details

The configuration returned by defcon is intended as a help to set up a syntactically correct configuration for the sam model. The dimensions are set from the data (years, age-classes, and fleet types available). The configuration is intended to be fairly simplistic in the hope that the model configured will at least converge (not guaranteed). Most importantly: No model validation has been performed, so it should not be assumed that the returned model configuration will result in a sensible assessment of the stock. The actual model configuration is the responsibility of the user.

Value

a list containing the elements needed to configure a sam model (e.g. minAge, maxAge, maxAgePlusGroup, keyLogFsta, ...).

`defpar` *Setup initial values for all model parameters and random effects.*

Description

Setup initial values for all model parameters and random effects.

Usage

```
defpar(dat, conf, spinoutyear = 10)
```

Arguments

`dat`	sam data object as returned from the function `setup.sam.data`
`conf`	sam configuration list, which could be read from a configuration file via the `loadConf` function. A default/dummy configuration can be generated via the `defcon` function.
`spinoutyear`	Technical setting only used for biological parameter process models to insure equilibrium distribution in final edge year

Details

The model parameters and random effects are not initialized in any clever way - most are simply set to zero. If convergence problems occour different initial values can be tested, but it is more likely a problem with the model configuration.

Value

a list containing initial values for all model parameters and random effects in the model.

deterministicReferencepoints

Function to calculate reference points for the embedded deterministic model of a SAM fit

Description

The function estimates reference points based on deterministic per-recruit calculations with no process variance. The following reference points are implemented:

F=x F fixed to x, e.g., "F=0.3"

StatusQuo F in the last year of the assessment

StatusQuo-y F in the y years before the last in the assessment, e.g., "StatusQuo-1"

MSY F that maximizes yield

0.xMSY Fs that gives 0.x*100% of MSY, e.g., "0.95MSY"

Max F that maximizes yield per recruit

0.xdYPR F such that the derivative of yield per recruit is 0.x times the derivative at F=0, e.g., "0.1dYPR"

0.xSPR F such that spawners per recruit is 0.x times spawners per recruit at F=0, e.g., "0.35SPR"

0.xB0 F such that biomass is 0.x times the biomass at F=0, e.g., "0.2B0"

Usage

```
deterministicReferencepoints(fit, referencepoints, ...)

## S3 method for class 'sam'
deterministicReferencepoints(
  fit,
  referencepoints,
  catchType = "catch",
  nYears = 100,
  Fsequence = seq(0, 2, len = 50),
  aveYears = max(fit$data$years) + (-9:0),
  selYears = max(fit$data$years),
  biasCorrect = FALSE,
  newton.control = list(),
  run = TRUE,
  equilibriumMethod = c("AD", "EC"),
  nosim_ci = 200,
  ncores = 1,
  ...
)
```

Arguments

`fit`	A fitted SAM model
`referencepoints`	list of reference points to calculate (See details)
`...`	other arguments not used
`catchType`	Type of yield to optimize: landing, catch, or discard
`nYears`	Number of years in per-recruit calculations
`Fsequence`	Sequence of F values for plotting and starting values
`aveYears`	Years to average over for biological input
`selYears`	Years to average over for selectivity
`biasCorrect`	Should bias correction be used in sdreport?
`newton.control`	Control arguments passed to the newton optimizer (See newton)
`run`	Run estimation? If false, a list of arguments to MakeADFun is returned.
`equilibriumMethod`	Method to use to find equilibrium
`nosim_ci`	Number of simulations for simulation based confidence intervals (only when equilibriumMethod is EC)
`ncores`	Number of cores for simulation

Value

List of estimated reference points

List of estimated reference points

Examples

```
## Not run:
 deterministicReferencepoints(fit, c("MSY","0.95MSY","Max","0.35SPR","0.1dYPR","StatusQuo-3"))

## End(Not run)
```

`empirobscorrplot` *Plots the residual between-age correlation matrices by fleet.*

Description

Plots the residual between-age correlation matrices by fleet.

Usage

```
empirobscorrplot(res, ...)

## S3 method for class 'samres'
empirobscorrplot(res, ...)
```

Arguments

`res`	the object returned from residuals.sam
`...`	extra arguments to plot

`equilibriumbiomassplot` *SAM equilibrium biomass plot*

Description

SAM equilibrium biomass plot

Usage

```
equilibriumbiomassplot(fit, ...)

## Default S3 method:
equilibriumbiomassplot(fit, ...)

## S3 method for class 'samforecast'
equilibriumbiomassplot(fit, ...)

## S3 method for class 'hcr'
equilibriumbiomassplot(fit, ...)
```

Arguments

`fit`	the object returned from sam.fit
`...`	extra arguments transferred to plot including the following: `add` logical, plotting is to be added on existing plot `ci` logical, confidence intervals should be plotted `cicol` color to plot the confidence polygon

Details

Plot of deterministic equilibrium spawners per recruit assuming biological parameters and selectivity for that year remains unchanged in the future.

equilibriumbiomasstable *equilibrium biomass table*

Description

equilibrium biomass table

Usage

```
equilibriumbiomasstable(fit, ...)

## Default S3 method:
equilibriumbiomasstable(fit, ...)
```

Arguments

`fit`	...
`...`	extra arguments not currently used

Details

...

erbplot *SAM effective reproductive output biomass plot*

Description

SAM effective reproductive output biomass plot

Usage

```
erbplot(fit, ...)

## Default S3 method:
erbplot(fit, ...)

## S3 method for class 'samforecast'
erbplot(fit, ...)

## S3 method for class 'hcr'
erbplot(fit, ...)
```

Arguments

`fit`	the object returned from sam.fit
`...`	extra arguments transferred to plot including the following: `add` logical, plotting is to be added on existing plot `ci` logical, confidence intervals should be plotted `cicol` color to plot the confidence polygon

Details

Plot of spawning stock biomass

`erbtable` *Effective reproductive biomass table*

Description

Effective reproductive biomass table

Usage

```
erbtable(fit, ...)

## Default S3 method:
erbtable(fit, ...)
```

Arguments

`fit`	...
`...`	extra arguments not currently used

Details

...

`faytable` *F-at-age table*

Description

F-at-age table

Usage

```
faytable(fit, ...)

## S3 method for class 'sam'
faytable(fit, fleet = which(fit$data$fleetTypes == 0), ...)
```

Arguments

`fit`	a fitted object of class 'sam' as returned from sam.fit
`...`	extra arguments not currently used
`fleet`	the fleet number(s) to return F summed for (default is to return the sum of all residual fleets).

Details

...

`fbarplot` *SAM Fbar plot*

Description

SAM Fbar plot

Usage

```
fbarplot(fit, ...)

## S3 method for class 'sam'
fbarplot(
  fit,
  partial = TRUE,
  drop = NULL,
  pcol = "lightblue",
  page = NULL,
  plot = TRUE,
  effectiveF = any(!fit$conf$seasonTimes %in% c(0, 1)),
  ...
)

## S3 method for class 'samset'
fbarplot(
  fit,
  partial = FALSE,
  drop = NULL,
  pcol = "lightblue",
  page = NULL,
  ...
)

## S3 method for class 'samforecast'
fbarplot(
  fit,
  partial = FALSE,
  drop = NULL,
  pcol = "lightblue",
  page = NULL,
  ...
)

## S3 method for class 'hcr'
fbarplot(
  fit,
  partial = FALSE,
  drop = NULL,
  pcol = "lightblue",
  page = NULL,
  ...
)
```

Arguments

`fit`	the object returned from sam.fit
`...`	extra arguments transferred to plot including the following: `add` logical, plotting is to be added on existing plot `ci` logical, confidence intervals should be plotted `cicol` color to plot the confidence polygon
`partial`	true if included partial F's are to be plotted
`drop`	number of years to be left unplotted at the end. Default (NULL) is to not show years at the end with no catch information
`pcol`	color of partial lines
`page`	partial ages to plot
`plot`	true if fbar should be plotted
`effectiveF`	If TRUE, effective full year F based on catch and survival is plotted. If FALSE, full year F based on survival is plotted.

Details

Plot the defined fbar.

`fbartable` *Fbar table*

Description

Fbar table

Usage

```
fbartable(fit, ...)

## Default S3 method:
fbartable(fit, ...)
```

Arguments

`fit` ...

`...` extra arguments not currently used

Details

...

`fitfromweb` *Read a fitted model from stockassessment.org*

Description

Read a fitted model from stockassessment.org

Usage

```
fitfromweb(stockname, character.only = FALSE, return.all = FALSE)
```

Arguments

`stockname` The short-form name of a stock on stockassessment.org. This will (currently?) not work for stocks defined via the AD Model builder version of SAM.

`character.only` a logical indicating whether ’stockname’ can be assumed to be a character string

`return.all` a logical indicating whether everything from model.RData should be returned in an environment

Details

...

fitplot *Plots fit to data*

Description

Plots fit to data

Usage

```
fitplot(fit, log = TRUE, ...)

## S3 method for class 'sam'
fitplot(fit, log = TRUE, fleets = unique(fit$data$aux[, "fleet"]), ...)
```

Arguments

fit	the object returned from sam.fit
log	should the plot be against log-obs
...	extra arguments to plot
fleets	an integer vector of fleets to plot. Default is all of them

forecast *forecast function to do shortterm*

Description

forecast function to do shortterm

Usage

```
forecast(
  fit,
  fscale = NULL,
  catchval = NULL,
  catchval.exact = NULL,
  fval = NULL,
  nextssb = NULL,
  landval = NULL,
  cwF = NULL,
  nosim = 1000,
  year.base = max(fit$data$years),
  ave.years = max(fit$data$years) + (-4:0),
  rec.years = max(fit$data$years) + (-9:0),
  label = NULL,
  overwriteSelYears = NULL,
  deterministic = FALSE,
  processNoiseF = TRUE,
  customWeights = NULL,
  customSel = NULL,
  lagR = FALSE,
  splitLD = FALSE,
  addTSB = FALSE,
  useSWmodel = (fit$conf$stockWeightModel >= 1),
  useCWmodel = (fit$conf$catchWeightModel >= 1),
  useMOmodel = (fit$conf$matureModel >= 1),
  useNMmodel = (fit$conf$mortalityModel >= 1),
  savesim = FALSE,
  cf.cv.keep.cv = matrix(NA, ncol = 2 * sum(fit$data$fleetTypes == 0), nrow =
    length(catchval)),
  cf.cv.keep.fv = matrix(NA, ncol = 2 * sum(fit$data$fleetTypes == 0), nrow =
    length(catchval)),
  cf.keep.fv.offset = matrix(0, ncol = sum(fit$data$fleetTypes == 0), nrow =
    length(catchval)),
  estimate = median
)
```

Arguments

`fit` an assessment object of type sam, as returned from the function sam.fit

`fscale` a vector of f-scales. See details.

`catchval` a vector of target catches. See details.

`catchval.exact` a vector of target catches which will be met without noise. See details.

`fval` a vector of target f values. See details.

`nextssb` a vector target SSB values the following year. See details

`landval` a vector of target catches. See details.

`cwF` a vector target custom weighted F values. customWeights must also be specified

`nosim` number of simulations default is 1000

`year.base` starting year default last year in assessment. Currently it is only supported to use last assessment year or the year before

`ave.years` vector of years to average for weights, maturity, M and such

`rec.years` vector of years to use to resample recruitment from

`label` optional label to appear in short table

`overwriteSelYears`
if a vector of years is specified, then the average selectivity of those years is used (not recommended)

`deterministic` option to turn all process noise off (not recommended, as it will likely cause bias)

`processNoiseF` option to turn off process noise in F `customWeights` a vector of same length as number of age groups giving custom weights (currently only used for weighted average of F calculation)

`customSel` supply a custom selection vector that will then be used as fixed selection in all years after the final assessment year (not recommended)

`lagR` if the second youngest age should be reported as recruits

`splitLD` if TRUE the result is split in landing and discards

`addTSB` if TRUE the total stock biomass (TSB) is added

`useSWmodel` if TRUE the catch mean weight predicted from the assessment model is used (can only be used for configurations supporting this)

`useCWmodel` if TRUE the catch mean weight predicted from the assessment model is used (can only be used for configurations supporting this)

`useMOmodel` if TRUE the proportion mature predicted from the assessment model is used (can only be used for configurations supporting this)

`useNMmodel` if TRUE the natural mortality predicted from the assessment model is used (can only be used for configurations supporting this)

`savesim` save the individual simulations

`cf.cv.keep.cv` exotic option

`cf.cv.keep.fv` exotic option

`cf.keep.fv.offset`
exotic option

`estimate` the summary function used (typically mean or median)

Details

There are three ways to specify a scenario. If e.g. four F values are specified (e.g. fval=c(.1,.2,.3,4)), then the first value is used in the last assessment year (base.year), and the three following in the three following years. Alternatively F's can be specified by a scale, or a target catch. Only one option can be used per year. So for instance to set a catch in the first year and an F-scale in the following one would write catchval=c(10000,NA,NA,NA), fscale=c(NA,1,1,1). The length of the vector specifies how many years forward the scenarios run.

Value

an object of type samforecast

`forecastMSY` *Estimating Fmsy*

Description

Estimating Fmsy

Usage

```
forecastMSY(
  fit,
  nYears = 100,
  nlminb.control = list(eval.max = 2000, iter.max = 2000),
  rec.years = c(),
  ave.years = max(fit$data$years) + (-9:0),
  processNoiseF = FALSE,
  ...
)

## S3 method for class 'sam'
forecastMSY(
  fit,
  nYears = 100,
  nlminb.control = list(eval.max = 2000, iter.max = 2000, trace = 1),
  rec.years = c(),
  ave.years = max(fit$data$years) + (-9:0),
  processNoiseF = FALSE,
  jacobianHScale = 0.5,
  nCatchAverageYears = 20,
  ...
)
```

Arguments

`fit` a SAM fit

`nYears` Number of years to forecast

`nlminb.control` list of control variables for nlminb

`rec.years` Numeric vector of years to use (to calculate mean and standard deviation) for recruitment. An empty vector will use the recruitment model.

`ave.years` vector of years to average for weights, maturity, M and such. Following ICES guidelines, the default is the last 10 years.

`processNoiseF` Should random walk process noise be used for F?

`...` other arguments passed to forecast

`jacobianHScale` Scale step size in jacobian calculation

`nCatchAverageYears`
Number of years to average catch over for finding MSY

References

Albertsen, C. M. and Trijoulet, V. (2020) Model-based estimates of reference points in an age-based state-space stock assessment model. Fisheries Research, 230, 105618. doi: 10.1016/j.fishres.2020.105618

See Also

forecast referencepoints

fselectivityplot *SAM F-selectivity plot*

Description

SAM F-selectivity plot

Usage

```
fselectivityplot(fit, cexAge = 1, ...)

## S3 method for class 'sam'
fselectivityplot(fit, cexAge = 1, ...)
```

Arguments

`fit`	An object returned from sam.fit
`cexAge`	cex variable giving the size of the age numbers
`...`	extra arguments transferred to barplot and text

Details

Plots selectivity in F.

generationlengthplot *SAM generation length plot*

Description

SAM generation length plot

Usage

```
generationlengthplot(fit, ...)

## Default S3 method:
generationlengthplot(fit, ...)

## S3 method for class 'samforecast'
generationlengthplot(fit, ...)

## S3 method for class 'hcr'
generationlengthplot(fit, ...)
```

Arguments

`fit` the object returned from sam.fit

`...` extra arguments transferred to plot including the following:
`add` logical, plotting is to be added on existing plot
`ci` logical, confidence intervals should be plotted
`cicol` color to plot the confidence polygon

Details

Plot of life expectancy

`generationlengthtable` *Generation length table*

Description

Generation length table

Usage

```
generationlengthtable(fit, ...)

## Default S3 method:
generationlengthtable(fit, ...)
```

Arguments

`fit` ...

`...` extra arguments not currently used

Details

...

`getAllDerivedValues` *Update sam fit with additional derived values*

Description

Update sam fit with additional derived values

Usage

```
getAllDerivedValues(fit)
```

Arguments

`fit`	sam fit returned by sam.fit

Value

Updated sam fit

`getFleet` *Extract a fleet observed or predicted value from a fitted object*

Description

Extract a fleet observed or predicted value from a fitted object

Usage

```
getFleet(fit, fleet, pred = "FALSE")
```

Arguments

`fit`	A fitted object as returned from sam.fit
`fleet`	The fleet number
`pred`	Should it be predicted value, default is observed

Details

Extract for example the observed or predicted catch at age of fleet "fleet"

Value

A matrix of observed or predicted values for fleet "fleet"

`getLowerBounds` *Bounds*

Description

Bounds

Usage

```
getLowerBounds(parameters, conf)
```

Arguments

`parameters`	initial values for the model in a format similar to what is returned from the defpar function
`conf`	model configuration in a format similar to what is returned from the defcon function

Value

a named list

`getResidualFleets` *Extract a list of catch fleets*

Description

Extract a list of catch fleets

Usage

```
getResidualFleets(fit, pred = "FALSE")
```

Arguments

`fit`	A fitted object as returned from sam.fit
`pred`	Should it be predicted value, default is observed

Value

A list of matrices of observed or predicted values for catch fleets

`getUpperBounds` *Bounds*

Description

Bounds

Usage

```
getUpperBounds(parameters, conf)
```

Arguments

`parameters`	initial values for the model in a format similar to what is returned from the defpar function
`conf`	model configuration in a format similar to what is returned from the defcon function

Value

a named list

grad *Calculate gradient of a function*

Description

Calculate gradient of a function

Usage

```
grad(
  func,
  x,
  h = abs(1e-04 * x) + 1e-04 * (abs(x) < sqrt(.Machine$double.eps/7e-07)),
  ...
)
```

Arguments

`func`	function
`x`	parameter values
`h`	step size
`...`	passed to func

Value

gradient vector

hcr *Harvest control rule forecast*

Description

The formula below is used to determine a new F based on the previous SSB.

$$F = \begin{cases} F_{cap} & SSB < B_{cap} \\ min\left(F_{target}, \max\left(F_{origin}, (SSB - B_{origin}) \cdot (F_{target} - F_{origin})/(B_{trigger} - B_{origin})\right)\right) & SSB \geq B_{cap} \end{cases}$$

If $B_{trigger} = B_{origin}$ and $SSB \geq B_{cap}$, F_{target} is always returned.

Usage

```
hcr(fit, ...)

## S3 method for class 'sam'
hcr(
  fit,
  nYears = 20,
  Ftarget,
  Btrigger,
  Forigin = 0,
  Borigin = 0,
  Fcap = 0,
  Bcap = 0,
  nosim = 10000,
  ave.years = max(fit$data$years) + (-4:0),
  rec.years = numeric(0),
  preForecast = list(),
  currentSSB = FALSE,
  ...
)
```

Arguments

`fit`	A SAM fit
`...`	additional arguments passed to modelforecast
`nYears`	Number of years to forecast
`Ftarget`	Target F for high SSB
`Btrigger`	SSB that triggers the control rule
`Forigin`	F used for SSB = Borigin
`Borigin`	Between Blim and Btrigger, F values are selected based on linear interpolation from Forigin to Ftarget
`Fcap`	F for SSB < Bcap
`Bcap`	SSB for which Fcap is used below
`nosim`	Number of simulations to do. If NULL a model forecast based on the Laplace approximation is used
`ave.years`	vector of years to average for weights, maturity, M and such
`rec.years`	vector of years to use to resample recruitment from. If an empty vector is given, recruitment is based on the fitted model.
`preForecast`	list of forecast parameters (i.e., fval, fscale, catchval, landval, or nextssb) to use before the HCR
`currentSSB`	if TRUE, SSB at the begining of the control rule year is used. If FALSE, SSB in the previous year is used. If any propF > 0, currentSSB must be FALSE.

Value

model forecast using a harvest control rule

hcr model forecast object

See Also

modelforecast

---

hessian *Calculate hessian of a function*

Description

Calculate hessian of a function

Usage

```
hessian(
  func,
  x,
  h = abs(1e-04 * x) + 1e-04 * (abs(x) < sqrt(.Machine$double.eps/7e-07)),
  columns = seq_along(x),
  ...
)
```

Arguments

func	function
x	parameter values
h	step size
columns	columns of hessian to calculate
...	passed to func

Value

jacobian matrix

Note

Could be made more accurate in some cases

ibc *Integrated spline basis for use with formula interface*

Description

Integrated spline basis for use with formula interface

Usage

```
ibc(x, df = 3L, knots = NULL, Boundary.knots = range(x), intercept = FALSE)
```

Arguments

x	Points to evaluate the basis in
df	Degrees of freedom
knots	Internal knots. If NULL, they are selected from quantiles of x.
Boundary.knots	Boundary knots. Defaults to range of x
intercept	Include an intercept in basis?

Value

A spline basis

icesAdviceRule *Forecast with an ICES advice rule*

Description

Forecast with an ICES advice rule

Usage

```
icesAdviceRule(
  x,
  Fmsy,
  MSYBtrigger,
  Blim,
  nosim = 10000,
  ave.years = max(x$data$years) + (-4:0),
  rec.years = numeric(0),
  preForecast = list(),
  currentSSB = FALSE,
  ...
)
```

Arguments

`x` Fitted assessment model

`Fmsy` ICES Fmsy which is used as target F

`MSYBtrigger` ICES MSYBtrigger below which F is reduced

`Blim` ICES Blim below which F is set to zero.

`nosim` Number of simulations to do. If NULL a model forecast based on the Laplace approximation is used

`ave.years` vector of years to average for weights, maturity, M and such

`rec.years` vector of years to use to resample recruitment from. If an empty vector is given, recruitment is based on the fitted model.

`preForecast` list of forecast parameters (i.e., fval, fscale, catchval, landval, or nextssb) to use before the HCR

`currentSSB` if TRUE, SSB at the begining of the control rule year is used. If FALSE, SSB at the begining of the previous year is used.

`...` Other arguments passes to hcr

Value

hcr object

Warning

The function does not make a short term forecast to see if fishing can continue below Blim.

References

ICES (2021) Advice on fishing opportunities. DOI: 10.17895/ices.advice.7720

See Also

hcr

---

iibc *Double integrated spline basis for use with formula interface*

Description

Double integrated spline basis for use with formula interface

Usage

```
iibc(x, df = 3L, knots = NULL, Boundary.knots = range(x), intercept = FALSE)
```

Arguments

`x`	Points to evaluate the basis in
`df`	Degrees of freedom
`knots`	Internal knots. If NULL, they are selected from quantiles of x.
`Boundary.knots`	Boundary knots. Defaults to range of x
`intercept`	Include an intercept in basis?

Value

A spline basis

`is.whole.positive.number` *Function to test if x is ...*

Description

Function to test if x is ...

Usage

```
is.whole.positive.number(x, tol = .Machine$double.eps^0.5)
```

Arguments

`x`	number
`tol`	precision

Details

...

jacobian *Calculate jacobian of a function*

Description

Calculate jacobian of a function

Usage

```
jacobian(
  func,
  x,
  h = abs(1e-04 * x) + 1e-04 * (abs(x) < sqrt(.Machine$double.eps/7e-07)),
  maxit = 30L,
  tol = 1e-12,
  subset = seq_along(x),
  ...
)
```

Arguments

func	function
x	parameter values
h	step size
maxit	maximum number of iterations
tol	tolerance
subset	subset indices of parameters to calculate jacobian wrt
...	passed to func

Value

jacobian matrix

jit *Jitter runs*

Description

Jitter runs

Usage

```
jit(fit, nojit = 10, ...)

## S3 method for class 'sam'
jit(
  fit,
  nojit = 10,
  par = defpar(fit$data, fit$conf),
  sd = 0.25,
  ncores = detectCores(),
  ...
)
```

Arguments

`fit` a fitted model object as returned from sam.fit

`nojit` a list of vectors. Each element in the list specifies a run where the fleets mentioned are omitted

`par` initial values to jitter around. The defaule ones are returned from the defpar function

`sd` the standard deviation used to jitter the initial values (most parameters are on a log scale, so similar to cv)

`ncores` the number of cores to attemp to use

Details

...

Value

A "samset" object, which is basically a list of sam fits

`leaveout` *leaveout run*

Description

leaveout run

Usage

```
leaveout(
  fit,
  fleet = as.list(2:fit$data$noFleets),
  ncores = detectCores(),
  ...
)
```

Arguments

`fit`	a fitted model object as returned from sam.fit
`fleet`	a list of vectors. Each element in the list specifies a run where the fleets mentioned are omitted
`ncores`	the number of cores to attemp to use
`...`	extra arguments to `sam.fit`

Details

...

`lifeexpectancyplot` *SAM life expectancy plot*

Description

SAM life expectancy plot

Usage

```
lifeexpectancyplot(fit, atRecruit = TRUE, ...)

## Default S3 method:
lifeexpectancyplot(fit, atRecruit = TRUE, ylimAdd = fit$conf$maxAge, ...)

## S3 method for class 'samforecast'
lifeexpectancyplot(fit, atRecruit = TRUE, ylimAdd = fit$conf$maxAge, ...)

## S3 method for class 'hcr'
lifeexpectancyplot(fit, atRecruit = TRUE, ylimAdd = fit$conf$maxAge, ...)
```

Arguments

`fit`	the object returned from sam.fit
`atRecruit`	If true, show life expectancy given survival until minAge, otherwise show life expectancy at birth
`...`	extra arguments transferred to plot including the following: `add` logical, plotting is to be added on existing plot `ci` logical, confidence intervals should be plotted `cicol` color to plot the confidence polygon
`ylimAdd`	values to add when calculating ylim for the plot

Details

Plot of life expectancy

lifeexpectancytable *Life expectancy table*

Description

Life expectancy table

Usage

```
lifeexpectancytable(fit, atRecruit = TRUE, ...)

## Default S3 method:
lifeexpectancytable(fit, atRecruit = TRUE, ...)
```

Arguments

`fit`	...
`atRecruit`	If true, show life expectancy given survival until minAge, otherwise show life expectancy at birth
`...`	extra arguments not currently used

Details

...

loadConf *Loads a model configuration from a file*

Description

Loads a model configuration from a file

Usage

```
loadConf(dat, file, patch = TRUE)
```

Arguments

`dat`	sam data list as returned from the function setup.sam.data
`file`	the file to read the configuration from
`patch`	logical if TRUE missing entries will be automatically filled with default values

Details

function useful loading a model configuration. Such a configuration can be saved via the saveConf function

`logLik.sam` *Log likelihood of sam object*

Description

Log likelihood of sam object

Usage

```
## S3 method for class 'sam'
logLik(object, ...)
```

Arguments

`object`	sam fitted object as returned from the `sam.fit` function
`...`	extra arguments

Details

log likelihood of a sam model run

`modelDescription` *Description of model*

Description

Description of model

Usage

```
modelDescription(fit, ...)
```

Arguments

`fit`	returned object from sam.fit
`...`	Additional parameters to be passed to ...

Details

...

modelforecast *Model based forecast function*

Description

Model based forecast function

Model based forecast function

Usage

```
modelforecast(fit, ...)

## S3 method for class 'sam'
modelforecast(
  fit,
  constraints = NULL,
  fscale = NULL,
  catchval = NULL,
  fval = NULL,
  nextssb = NULL,
  landval = NULL,
  nosim = 0,
  year.base = max(fit$data$years),
  ave.years = max(fit$data$years) + (-9:0),
  rec.years = c(),
  label = NULL,
  overwriteSelYears = NULL,
  deterministicF = FALSE,
  processNoiseF = FALSE,
  fixedFdeviation = FALSE,
  useFHessian = FALSE,
  resampleFirst = !is.null(nosim) && nosim > 0,
  useModelLastN = TRUE,
  customSel = NULL,
  lagR = FALSE,
  splitLD = FALSE,
  addTSB = FALSE,
  biasCorrect = FALSE,
  returnAllYears = FALSE,
  returnObj = FALSE,
  progress = TRUE,
  estimate = median,
  silent = TRUE,
  newton_config = NULL,
  custom_pl = NULL,
  useNonLinearityCorrection = (nosim > 0 && !deterministicF),
  ncores = 1,
  ...
)
```

Arguments

`fit` SAM model fit

`...` other variables used by the methods

`constraints` a character vector of forecast constraint specifications

`fscale` a vector of f-scales. See details.

`catchval` a vector of target catches. See details "old specification".

`fval` a vector of target f values. See details "old specification".

`nextssb` a vector target SSB values the following year. See details "old specification".

`landval` a vector of target catches. See details "old specification".

`nosim` number of simulations. If 0, the Laplace approximation is used for forecasting.

`year.base` starting year default last year in assessment. Currently it is only supported to use last assessment year or the year before

`ave.years` vector of years to average for weights, maturity, M and such

`rec.years` vector of years to use to resample recruitment from. If the vector is empty, the stock recruitment model is used.

`label` optional label to appear in short table

`overwriteSelYears` if a vector of years is specified, then the average selectivity of those years is used (not recommended)

`deterministicF` option to set F variance to (almost) zero (not recommended)

`processNoiseF` option to turn off process noise in F

`fixedFdeviation` Use a fixed F deviation from target?

`useFHessian` Use the covariance of F estimates instead of the estimated process covariance for forecasting?

`resampleFirst` Resample base year when nosim > 0?

`useModelLastN` Use last N?

`customSel` supply a custom selection vector that will then be used as fixed selection in all years after the final assessment year (not recommended)

`lagR` if the second youngest age should be reported as recruits

`splitLD` if TRUE the result is split in landing and discards

`addTSB` if TRUE the total stock biomass (TSB) is added

`biasCorrect` Do bias correction of reported variables. Can be turned off to reduce running time (not recommended).

`returnAllYears` If TRUE, all years are bias corrected. Otherwise, only forecast years are corrected.

`returnObj` Only return TMB object?

`progress` Show progress bar for simulations?

`estimate` the summary function used (typically mean or median) for simulations

`silent` Passed to MakeADFun. Should the TMB object be silent?

`newton_config` Configuration for newton optimizer to find F values. See ?TMB::newton for details. Use NULL for TMB defaults.

`custom_pl` Parameter list. By default, the parameter list from fit is used.

`useNonLinearityCorrection`
Should a non linearity correction be added to transformation of logF? See Details - Non-linearity correction.

`ncores` Number of cores to use if simulating

Details

Function to forecast the model under specified catch constraints. In the forecast, catch constraints are used to set the mean of the $log(F)$ process for each simulation. Therefore, catch constraints are not matched exactly in individual simulations. Likewise, the summary of a specific set of simulations will not match exactly due to random variability. By default, recruitment is forecasted using the estimated recruitment model. If a vector of recruitment years is given, recruitment is forecasted using a log-normal distribution with the same mean and variance as the recruitment in the years given. This is different from the forecast function, which samples from the recruitment estimates. Catch scenarios are specified by a vector of target constraints. The first value determines F in the year after the base year.

Value

an object of type samforecast

Forecast constraints

F based constraints:: Forecasts for F values are specified by the format "F[f,a0-a1]=x" where f is the residual catch fleet and a0-a1 is an age range. For example, "F[2,2-4]=0.3" specifies that the average F for the second fleet over ages 2-4 should be 0.3. If an "*" is added to the target value, the target will be relative to the year before. For example, "F[2,2-4]=0.9*" specifies that the average F for the second fleet over ages 2-4 should be 90 If the fleet is omitted (e.g., F[2-4]), the target is for the total F. If the age range is omitted (e.g., F[2]), the fbar range of the model is used. Likewise, both fleet and age range can be omited (e.g., F=0.3) to specify a value for total F with the range used in the model.

For example:

"F=0.2" Will set the median average total fishing mortality rate to 0.2

"F[1 =0.2"] Will set the median average fishing mortality rate of the first fleet to 0.2

"F[2-4 =0.2"] Will set the median average total fishing mortality rate over ages 2 to 4 to 0.2

"F[3,2-4 =0.2"] Will set the median average fishing mortality rate over ages 2 to 4 for the third fleet to 0.2

Catch/Landing based constraints:: Forecasts for catch and landing values are specified by the format "C[f,a0-a1]=x" for catch and "L[f,a0-a1]" for landings. If the age range is omitted, all modelled ages are used. Otherwise, the format is similar to F based scenarios. If an "*" is added to the target value, the target will be relative to the year before. Further, the catch target for a fleet can be relative to the total by adding "*C" or to another fleet by adding "*C[f]" where f is the fleet number. The same age range will always be used. Likewise, relative landing targets can be specified using "*", "*L", or "*L[f]" for targets relative to last year, the total, or fleet f, respectively.

For example:

"C=100000" Will scale F such that the total predicted catch is 100000

"C[1 =100000"] Will scale F such that the predicted catch of the first fleet is 100000

"C[2-4 =100000"] Will scale F such that the total predicted catch for ages 2 to 4 is 100000

"C[3,2-4 =100000"] Will scale F such that the predicted catch for ages 2 to 4 in the third fleet is 100000

"L=100000" Will scale F such that the total predicted landing is 100000

"L[1 =100000"] Will scale F such that the predicted landing of the first fleet is 100000

"L[2-4 =100000"] Will scale F such that the total predicted landing for ages 2 to 4 is 100000

"L[3,2-4 =100000"] Will scale F such that the predicted landing for ages 2 to 4 in the third fleet is 100000

Next year's SSB/TSB based constraints:: Forecasts for spawning stock biomass (SSB) and total stock biomass (TSB) values are specified by the format "SSB[a0-a1]=x" for SSB and "TSB[a0-a1]" for TSB. For setting F in year y, the relevant biomass for year y+1 is predicted for the constraint. If spawning is not at the beginning of the year, F is assumed to be the same for year y and y+1 in the prediction. The format is similar to catch/landing based scenarios. However, fleets have no effect. If an age range is omitted, the full age range of the model is used. If an "*" is added to the target value, the target will be relative to the year before. That is, when setting F in year y, the predicted biomass in year y+1 will be relative to the biomass in year y-1. Note that since SSB and TSB used for catch constraints are predicted, the input constraint will differ from the output SSB and TSB estimates due to process variability.

For example:

SSB=200000 Will scale F such that the predicted SSB at the beginning of the next year is 200000

SSB[3-9 =200000] Will scale F such that the predicted SSB for ages 3 to 9 at the beginning of the next year is 200000

TSB=200000 Will scale F such that the predicted TSB at the beginning of the next year is 200000

TSB[3-9 =200000] Will scale F such that the predicted TSB for ages 3 to 9 at the beginning of the next year is 200000

Harvest control rule based constraints:: Harvest control rules can be specified for forecasts using the format "HCR=x~y" where x is the target and y is the biomass trigger (see ?hcr for full details on the form of the harvest control rule). Further, the target can be specified as an F target ("HCR=xF~y"), catch target ("HCR=xC~y"), or landing target ("HCR=xL~y"). Likewise the trigger can either be for SSB ("HCR=x~ySSB") or TSB ("HCR=x~yTSB"). Age ranges can be set for both triggers and targets and a fleet can be set for the target. The notation and defaults are similar to the F based and SSB/TSB based constraints, respectively. When setting F in year y, the projected biomass in year y is used by default. To use the (at this time known) biomass in a previous year, a time lag can be specified. To specify a time lag of, e.g., 1 year for SSB the format is "HCR=x~ySSB-1". Finally, the origin and cap for the HCR can be set using "HCR[FO=a,FC=b,BO=d,BC=e]=x~y", where FO is the F (or catch or landing) value at origin, BO is the biomass at origin, FC is the F (or catch or landing) value when the HCR is capped and BC is the biomass at which the HCR is capped. See ?hcr for further details on the shape of the HCR. For a HCR similar to the ICES advice rule, the specification is on the form "HCR[BC=Blim] = fmsy~MSYBtrigger". Note that, unlike an ICES advice rule, the HCR does not do a forecast to determine if fishing can continue below Blim.

For example:

HCR=0.9~100000 Will apply a harvest control rule with an F target of 0.9 and a biomass trigger of 100000 on SSB

HCR=10000C~100000 Will apply a harvest control rule with a catch target of 10000 and a biomass trigger of 100000 on SSB

HCR=0.9~100000SSB Will apply a harvest control rule with an F target of 0.9 and a biomass trigger of 100000 on SSB

HCR=0.9F[1,2-4 ~100000SSB] Will apply a harvest control rule with an F target on the first fleet ages 2-4 of 0.9 and a biomass trigger of 100000 on SSB

HCR=0.9~100000TSB[0-4] Will apply a harvest control rule with an F target of 0.9 and a biomass trigger of 100000 on TSB for ages 0 to 4

HCR[FC=1e-9,BC=20000 =0.9~100000] Will apply a harvest control rule with an F target of 0.9 and a biomass trigger of 100000 on SSB where biomass values below 20000 will give an F of 1e-9

HCR[FO=0,BO=30000 =0.9~100000] Will apply a harvest control rule with an F target of 0.9 and a biomass trigger of 100000 on SSB where the slope on which F is reduced goes to zero F at a biomass of 30000

Combining constraints:: Constraints for different fleets can be combined by "&". For example, "F[2-4]=0.5 & C[2]=10000" specifies that total Fbar over ages 2-4 should be 0.5 while the catch for the second residual catch fleet should be 10,000t. The constraints cannot affect within-fleet selectivity. Therefore, a fleet can at most have one constraint per year, and the total number of constraints cannot exceed the number of catch fleets. That is, if a constraint is given for the sum of fleets, there must be at least one fleet without any constraints. For fleets where no constraints are given, a constraint is set to keep their relative Fs constant.

Values relative to previous year:: Catch constraints specified as specific values are inherently different from catch constraints specified as relative values, even if they lead to the same F. Catch constraints specified as relative values will propagate the uncertainty in, e.g, F from previous years whereas constraints specified as specific values will not. This is different from the forecast function where, for example, a forecast using fval is the same as a forecast using fscale, if they lead to the same F.

##'

Process variability:: In the forecast, constraints are used to set the predicted F value in year y based on information available until year y-1. Therefore, constraints using predicted values for year y, such as catch, will not be matched exactly by the realized catch due to process variability in F, N, biological processes and catch itself.

Non-linearity correction

In the model forecasts, constraints are calculated to set the mean of the log(F) process, corresponding to the median F-at-ages. Typically, the constraints are non-linear functions of log(F)-at-age. Therefore, when stochasticity is added to log(F) (i.e., deterministicF=FALSE), target values will correspond to a transformation of the median, and not the median of the transformation. For example, a target for the average fishing mortality (Fbar) will correspond to the average of the median F at age, which will be different from the median Fbar.

The "useNonLinearityCorrection" argument can be used to shift the target from a function of the mean log(F) (median F) towards the log-mean of the function of log(F), which is approximately the median of the function of log(F).

Old specification

It is also possible to specify forecast constraints in a way similar to the forecast function. There are four ways to specify a scenario. If e.g. four F values are specified (e.g. fval=c(.1,.2,.3,4)), then the first value is used in the year after the last assessment year (base.year + 1), and the three following in the three following years. Alternatively F's can be specified by a scale, or a target catch. Only one option can be used per year. So for instance to set a catch in the first year and an F-scale in the following one would write catchval=c(10000,NA,NA,NA), fscale=c(NA,1,1,1). If only NA's are specified in a year, the F model is used for forecasting. The length of the vector specifies how many years forward the scenarios run. Unlike the forecast function, no value should be given for the base year. Internally, the old specification is translated such that "fval=x" becomes "F=x", "fscale=x" becomes "F=x*", "catchval=x" becomes "C=x", "nextssb=x" becomes "SSB=x", and "landval=x" becomes "L=x".

Forecasts using Laplace approximation or simulations

Forecasts can be made using either a Laplace approximation projection (nosim=0) or simulations (nosim > 0). When using the Laplace approximation, the most likely projected trajectory of the processes along with a confidence interval is returned. In contrast, simulation based forecasts will return individual simulated trajectories and summarize using the function given as the estimate argument along with an interval covering 95

Warnings

Long term forecasts with random walk recruitment can lead to unstable behaviour and difficulties finding suitable F values for the constraints. If no suitable F value can be found, an error message will be shown, and F values will be NA or NaN. Likewise, forecasts leading to high F values in some years (or large changes from one year to another) may cause problems for the optimization as they will be used as starting values for the next years. Since the model works on log space, all target values should be strictly positive. Values too close to zero may cause problems.

See Also

forecast

`modeltable` *model table*

Description

model table

Usage

```
modeltable(fits, ...)

## S3 method for class 'sam'
modeltable(fits, ...)

## S3 method for class 'samset'
modeltable(fits, ...)
```

Arguments

`fits`	A sam fit as returned from the sam.fit function, or a collection c(fit1, fit2, ...) of such fits
`...`	extra arguments not currently used

Details

...

`modelVersionInfo` *Description of model*

Description

Description of model

Usage

```
modelVersionInfo(fit, ...)
```

Arguments

`fit`	returned object from sam.fit
`...`	Additional parameters to be passed to ...

Details

Writes a string to install the version of the package which was used to run the model.

mohn *Mohn's rho calculation*

Description

Mohn's rho calculation

Usage

```
mohn(fits, what = NULL, lag = 0, ...)

## S3 method for class 'samset'
mohn(fits, what = NULL, lag = 0, ...)
```

Arguments

fits	a samset object as returned from the retro function.
what	a function computing the quantity to calculate Mohn's rho for (default NULL computes Fbar, SSB, and R).
lag	lag applied to fits and reference fit.
...	extra arguments

Details

...

MSE *Management strategy evaluation using SAM models*

Description

Management strategy evaluation using SAM models

Usage

```
MSE(
  OM,
  EM,
  nYears,
  AdviceForecastSettings,
  AdviceYears = 1,
  AdviceLag = 0,
  initialAdvice = NA,
  implementationError = function(x) x,
  knotRange = 3,
  intermediateFleets = numeric(0),
  OMselectivityFixed = FALSE,
  ...
)
```

Arguments

`OM`	sam.fit that will work as operating model
`EM`	sam.fit that will work as estimation model
`nYears`	Number of years to run simulation
`AdviceForecastSettings`	Settings to do forecast that determines advice
`AdviceYears`	Number of years advice given at a time. How advice is given is determined by AdviceForecastSettings
`AdviceLag`	Lag between assessment and advice
`initialAdvice`	Advice in the first AdviceLag years
`implementationError`	Function to add implementation error (i.e, transform advice to target catch)
`knotRange`	Range of spline knot values to try
`intermediateFleets`	Fleets that are available in the (first) intermediate year
`OMselectivityFixed`	Fix selectivity in OM?
`...`	arguments passed on to addSimulatedYears

Value

a list with MSE result

`nobs.sam` *Extract number of observations from sam object*

Description

Extract number of observations from sam object

Usage

```
## S3 method for class 'sam'
nobs(object, ...)
```

Arguments

`object`	sam fitted object as returned from the `sam.fit` function
`...`	extra arguments

Details

...

`nscodConf` *nscodConf*

Description

nscodConf

Usage

```
data("nscodConf")
```

Format

The format is: $ minAge $ maxAge $ maxAgePlusGroup $ keyLogFsta $ corFlag $ keyLogFpar $ keyQpow $ keyVarF $ keyVarLogN $ keyVarObs $ stockRecruitmentModelCode $ noScaledYears $ keyScaledYears $ keyParScaledYA $ fbarRange

Details

...

Source

...

References

...

Examples

```
data(nscodConf)
## maybe str(nscodConf) ; plot(nscodConf) ...
```

nscodData *nscodData*

Description

nscodData

Usage

```
data("nscodData")
```

Format

The format is: $ noFleets $ fleetTypes $ sampleTimes $ noYears $ years $ nobs $ idx1 $ idx2 $ aux $ logobs $ propMat $ stockMeanWeight $ catchMeanWeight $ natMor $ landFrac $ disMeanWeight $ landMeanWeight $ propF $ propM

Details

...

Source

...

References

...

Examples

```
data(nscodData)
## maybe str(nscodData) ; plot(nscodData) ...
```

nscodParameters *nscodParameters*

Description

nscodParameters

Usage

```
data("nscodParameters")
```

Format

The format is: List of 14 $ logFpar $ logQpow $ logSdLogFsta $ logSdLogN $ logSdLogObs $ rec_loga $ rec_logb $ itrans_rho $ logScale $ logScaleSSB $ logPowSSB $ logSdSSB $ logF $ logN

Details

...

Source

...

References

...

Examples

```
data(nscodParameters)
## maybe str(nscodParameters) ; plot(nscodParameters) ...
```

---

ntable *N table*

Description

N table

Usage

```
ntable(fit, ...)

## S3 method for class 'sam'
ntable(fit, ...)
```

Arguments

`fit`	...
`...`	extra arguments not currently used

Details

...

obscorrplot *Plots the estimated correlation matrices by fleet.*

Description

Plots the estimated correlation matrices by fleet.

Usage

```
obscorrplot(fit, ...)

## S3 method for class 'sam'
obscorrplot(fit, ...)
```

Arguments

`fit`	the object returned from sam.fit
`...`	extra arguments to plot

obscov *Extract observation covariance matrices from a SAM fit*

Description

Extract observation covariance matrices from a SAM fit

Usage

```
obscov(fit, corr = FALSE, ...)

## S3 method for class 'sam'
obscov(fit, corr = FALSE, ...)

## S3 method for class 'samset'
obscov(fit, corr = FALSE, ...)
```

Arguments

`fit`	the object returned from sam.fit
`corr`	if TRUE return correlation matrices rather than covariances
`...`	extra arguments not currently used

Value

a list of matrices

parplot *SAM parameter plot*

Description

SAM parameter plot

Usage

```
parplot(fit, cor.report.limit = 0.95, ...)

## S3 method for class 'sam'
parplot(fit, cor.report.limit = 0.95, ...)

## S3 method for class 'samset'
parplot(fit, cor.report.limit = 0.95, ...)
```

Arguments

`fit`	the object returned from sam.fit
`cor.report.limit`	correlations with absolute value > this number is reported in the plot
`...`	extra arguments transferred to plot

Details

Plot of all estimated model parameters (fixed effects). Shown with confidence interval.

partable *parameter table*

Description

parameter table

Usage

```
partable(fit, ...)

## S3 method for class 'sam'
partable(fit, ...)
```

Arguments

`fit`	...
`...`	extra arguments not currently used

Details

...

`plot.hcr` *Plot hcr object*

Description

Plot hcr object

Usage

```
## S3 method for class 'hcr'
plot(x, ...)
```

Arguments

`x`	output from the `hcr` function
`...`	extra arguments

Details

...

`plot.sam` *Plot sam object*

Description

Plot sam object

Usage

```
## S3 method for class 'sam'
plot(x, ...)
```

Arguments

`x`	fitted object as returned from the `sam.fit` function.
`...`	extra arguments (not possible to use add=TRUE — instead collect to a list of fits using e.g the c(...), and then plot that collected object).

Details

gives a 3 plot overview plot og ssb, fbar, and recruits. These plots are available individually via the functions `ssbplot`, `fbarplot`, and `recplot`.

`plot.samforecast` *Plot samforecast object*

Description

Plot samforecast object

Usage

```
## S3 method for class 'samforecast'
plot(x, ...)
```

Arguments

`x`	fitted object as returned from the `sam.fit` function
`...`	extra arguments

Details

...

`plot.samres` *Plot sam residuals*

Description

Plot sam residuals

Usage

```
## S3 method for class 'samres'
plot(x, type = "bubble", ...)
```

Arguments

`x`	an object of type 'samres' as returned from functions `residuals.sam` or `procres`.
`type`	either "bubble" (default) or "summary"
`...`	extra arguments

Details

In the "bubble" type red indicate negative residuals and blue positive. The area of the circles scales with the absolute size of the residuals.

Examples

```
## Not run:
data(nscodData)
data(nscodConf)
data(nscodParameters)
fit <- sam.fit(nscodData, nscodConf, nscodParameters)
par(ask=FALSE)
plot(residuals(fit))

## End(Not run)
```

---

plot.samset *Plot sam object*

Description

Plot sam object

Usage

```
## S3 method for class 'samset'
plot(x, ...)
```

Arguments

x	fitted object as returned from the sam.fit function.
...	extra arguments

Details

...

---

plot.samypr *Plot sam object*

Description

Plot sam object

Usage

```
## S3 method for class 'samypr'
plot(x, ...)
```

Arguments

`x`	...
`...`	extra arguments

Details

...

`plotby` *Plot by one or two*

Description

Plot by one or two

Usage

```
plotby(
  x = NULL,
  y = NULL,
  z = NULL,
  x.line = NULL,
  y.line = NULL,
  z.line = NULL,
  by = NULL,
  bubblescale = 1,
  x.common = !is.null(x),
  y.common = !is.null(y),
  z.common = !is.null(z),
  xlab = NULL,
  ylab = NULL,
  xlim = NULL,
  ylim = NULL,
  zmax = NULL,
  axes = TRUE,
  ...
)
```

Arguments

`x`	numeric vector of points to be plotted
`y`	numeric vector of points to be plotted
`z`	numeric vector of points to be plotted
`x.line`	numeric vector of points of line to be added
`y.line`	numeric vector of points of line to be added

`z.line`	numeric vector of points of line to be added
`by`	vector or two column matrix to create sub sets from
`bubblescale`	scaling of bubble size
`x.common`	logical: use same x-axis for all plots
`y.common`	logical: use same y-axis for all plots
`z.common`	logical: use same z-axis for all plots
`xlab`	normal graphical parameter
`ylab`	normal graphical parameter
`xlim`	normal graphical parameter
`ylim`	normal graphical parameter
`zmax`	internally used to scale bubbles similarly
`axes`	normal graphical parameter
`...`	additional graphical parameters

Details

Function used for splitting plots e.g. used to plot residuals

Examples

```
exdat<-expand.grid(age=1:5, year=1950:2016, fleet=1:3)
exdat$perfectres<-rnorm(nrow(exdat))
attach(exdat)
par(ask=FALSE)
plotby(year,age,perfectres, by=fleet)
detach(exdat)
```

`plotit` *Plot helper*

Description

Plot helper

Usage

```
plotit(fit, what, ...)

## S3 method for class 'sam'
plotit(
  fit,
  what,
  x = fit$data$years,
  ylab = what,
  xlab = "Years",
  ex = numeric(0),
  trans = function(x) x,
  add = FALSE,
  ci = TRUE,
  cicol = gray(0.5, alpha = 0.5),
  addCI = NA,
  drop = 0,
  unnamed.basename = "current",
  xlim = NULL,
  ylim = NULL,
  ylimAdd = NA,
  ...
)

## S3 method for class 'samset'
plotit(
  fit,
  what,
  x = fit$data$years,
  ylab = what,
  xlab = "Years",
  ex = numeric(0),
  trans = function(x) x,
  add = FALSE,
  ci = TRUE,
  cicol = gray(0.5, alpha = 0.5),
  addCI = rep(FALSE, length(fit)),
  drop = 0,
  unnamed.basename = "current",
  xlim = NULL,
  ...
)

## S3 method for class 'samforecast'
plotit(
  fit,
  what,
  x = fit$data$years,
  ylab = what,
  xlab = "Years",
  ex = numeric(0),
  trans = function(x) x,
  add = FALSE,
  ci = TRUE,
  cicol = gray(0.5, alpha = 0.5),
  addCI = NA,
  drop = 0,
  unnamed.basename = "current",
  xlim = NULL,
  ylim = NULL,
  ...
)

## S3 method for class 'hcr'
plotit(
  fit,
  what,
  x = fit$data$years,
  ylab = what,
  xlab = "Years",
  ex = numeric(0),
  trans = function(x) x,
  add = FALSE,
  ci = TRUE,
  cicol = gray(0.5, alpha = 0.5),
  addCI = NA,
  drop = 0,
  unnamed.basename = "current",
  xlim = NULL,
  ...
)
```

Arguments

`fit`	the fitted object from sam.fit of a set of such fits c(fit1,fit2)
`what`	quoted name of object to extract
`...`	extra arguments transferred to plot
`x`	x-values
`ylab`	label on y-axis
`xlab`	label on x-axis
`ex`	extra y's to make room for
`trans`	function to transform values by
`add`	logical, plotting is to be added on existing plot
`ci`	logical, confidence intervals should be plotted
`cicol`	color to plot the confidence polygon
`addCI`	A logical vector indicating if confidence intervals should be plotted for the added fits.
`drop`	number of years to be left unplotted at the end.
`unnamed.basename`	the name to assign an unnamed basefit
`xlim`	xlim for the plot
`ylim`	ylim for the plot
`ylimAdd`	values to add when calculating ylim for the plot

Details

The basic plotting used bu many of the plotting functions (e.g. ssbplot, fbarplot ...)

---

predstdplot *Prediction-standard deviation plot*

Description

Prediction-standard deviation plot

Usage

```
predstdplot(
  fit,
  fleet,
  age = NULL,
  type = "log",
  ylim = NULL,
  ylab = "Standard deviation",
  xlab = "Prediction",
  main = "Pred-std relation",
  ...
)
```

Arguments

fit	A sam fit object returned from sam.fit.
fleet	Fleet number to plot relation between prediction and standard deviation.
age	Relation at age. Only used in cases with more than one relation within the same fleet.
type	Either 'log' or 'natural': relation for observations on a log or natural scale.
ylim	Optional, sent to plot
ylab	Optional, sent to plot
xlab	Optional, sent to plot
main	Optional, sent to plot
...	Sent to plot

Details

Plot the relation between observation prediction and standard deviation.

print.hcr *Print hcr object*

Description

Print hcr object

Usage

```
## S3 method for class 'hcr'
print(x, ...)
```

Arguments

x	a sam hcr object as returned by hcr
...	extra arguments

Details

prints the HCR forecast

print.sam *Print sam object*

Description

Print sam object

Usage

```
## S3 method for class 'sam'
print(x, ...)
```

Arguments

x	the fitted object as returned from the sam.fit function
...	extra arguments

Details

prints the log-likelihood and the main convergence criteria

print.samcoef *Print samcoef object*

Description

Print samcoef object

Usage

```
## S3 method for class 'samcoef'
print(x, ...)
```

Arguments

x	samcoef object as returned from the coef.sam function
...	extra arguments

print.samforecast *Print samforecast object*

Description

Print samforecast object

Usage

```
## S3 method for class 'samforecast'
print(x, ...)
```

Arguments

x	an object as returned from the forecast function
...	extra arguments

Details

...

`print.samres` *Print samres object*

Description

Print samres object

Usage

```
## S3 method for class 'samres'
print(x, ...)
```

Arguments

`x`	a sam residual object as returned from either `residuals.sam` or `procres`
`...`	extra arguments

Details

prints the residuals as a data.frame

`print.samset` *Print samset object*

Description

Print samset object

Usage

```
## S3 method for class 'samset'
print(x, ...)
```

Arguments

`x`	a list of sam models
`...`	extra arguments

Details

...

print.samypr *Print samypr object*

Description

Print samypr object

Usage

```
## S3 method for class 'samypr'
print(x, ...)
```

Arguments

x	an object as returned from the ypr function
...	extra arguments

Details

...

print.sam_referencepoints *Print referencepoint object*

Description

Print referencepoint object

Usage

```
## S3 method for class 'sam_referencepoints'
print(x, tables = c("F", "Biomass", "Yield"), digits = 4, format = "f", ...)
```

Arguments

x	a sam referencepoint object as returned by referencepoints
tables	tables to print
digits	number of digits to print
format	printing format for numbers
...	extra arguments

Details

prints the F reference point table

procres — *Compute process residuals (single joint sample)*

Description

Compute process residuals (single joint sample)

Usage

```
procres(fit, map = fit$obj$env$map, ...)
```

Arguments

`fit`	the fitted object as returned from the `sam.fit` function
`map`	map from original fit
`...`	extra arguments (not currently used)

Details

Single joint sample residuals of log(F) and log(N)

Value

an object of class `samres`

qtable — *table of survey catchabilities*

Description

table of survey catchabilities

Usage

```
qtable(fit, ...)
```

Arguments

`fit`	...
`...`	extra arguments not currently used

Details

...

qtable.sam *table of survey catchabilities*

Description

table of survey catchabilities

Usage

```
## S3 method for class 'sam'
qtable(fit, ...)
```

Arguments

fit	A sam fit as returned from the sam.fit function
...	extra arguments not currently used

qtableplot *plot survey catchabilities*

Description

plot survey catchabilities

plot survey catchabilities

Usage

```
qtableplot(qt, exp = FALSE)

## S3 method for class 'samqtable'
qtableplot(qt, exp = FALSE)
```

Arguments

qt	An object of class 'samqtable' as returned from qtable
exp	if true return on natural scale rather than log

read.data.files — *Read all standard data SAM files and return a list as created by 'setup.sam.data'*

Description

Read all standard data SAM files and return a list as created by 'setup.sam.data'

Usage

```
read.data.files(dir = ".")
```

Arguments

`dir`	Directory to read from

Details

Read all standard SAM data files

Value

list (as created by 'setup.sam.data')

read.ices — *Function to read ICES/CEFAS data files and validate if input makes sense*

Description

Function to read ICES/CEFAS data files and validate if input makes sense

Usage

```
read.ices(filen)
```

Arguments

`filen`	The filename

Details

First two lines are ignored and can be used for comments. Can read formats 1 full, 2 row, 3 scalar, and 5 column

Tests: Formatcode is valid, years and ages are pos. integers minimum <= maximum for years and ages number of rows and coulmns match year and age ranges data contains only numbers.

Returns: A validated data matrix.

`read.surveys` *Function to read ices survey format*

Description

Function to read ices survey format

Usage

```
read.surveys(filen)
```

Arguments

`filen`	the file

Details

...

`read.table.nowarn` *Function to supress incomplete final line warning*

Description

Function to supress incomplete final line warning

Usage

```
read.table.nowarn(...)
```

Arguments

`...`	arguments

Details

...

recplot *SAM Recruits plot*

Description

SAM Recruits plot

Usage

```
recplot(fit, lagR = FALSE, ...)

## S3 method for class 'sam'
recplot(fit, lagR = FALSE, ...)

## S3 method for class 'samset'
recplot(fit, lagR = FALSE, ...)

## S3 method for class 'samforecast'
recplot(fit, lagR = FALSE, ...)

## S3 method for class 'hcr'
recplot(fit, lagR = FALSE, ...)
```

Arguments

`fit`	the object returned from sam.fit
`lagR`	use the age after the youngest as R
`...`	extra arguments transferred to plot including the following: `add` logical, plotting is to be added on existing plot `ci` logical, confidence intervals should be plotted `cicol` color to plot the confidence polygon

Details

Plot of numbers of recruits (youngest age class)

rectable *Recruit table*

Description

Recruit table

Usage

```
rectable(fit, lagR = FALSE, ...)

## Default S3 method:
rectable(fit, lagR = FALSE, ...)
```

Arguments

fit	...
lagR	use the age after the youngest as R
...	extra arguments not currently used

Details

...

reduce *reduce helper function to reduce data*

Description

reduce helper function to reduce data

Usage

```
reduce(data, year = NULL, fleet = NULL, age = NULL, conf = NULL)
```

Arguments

data	a data object as returned by the function setup.sam.data
year	a vector of years to be excluded.
fleet	a vector of fleets to be excluded.
age	a vector of ages fleets to be excluded.
conf	an optional corresponding configuration to be modified along with the data change. Modified is returned as attribute "conf"

Details

When more than one vector is supplied they need to be of same length, as only the pairs are excluded

`referencepoints` *Estimate reference points*

Description

Estimate reference points

Usage

```
referencepoints(
  fit,
  nYears,
  Fsequence,
  aveYears,
  selYears,
  SPRpercent,
  catchType,
  MSYreduction,
  newtonSteps = 3,
  optN = 100,
  jacobianHScale = 0.5,
  ...
)

## S3 method for class 'sam'
referencepoints(
  fit,
  nYears = 100,
  Fsequence = seq(0, 4, len = 200),
  aveYears = max(fit$data$years) + (-9:0),
  selYears = max(fit$data$years),
  SPRpercent = c(0.35),
  dYPRpercent = c(0.1),
  B0percent = c(0.2),
  catchType = "catch",
  MSYreduction = c(0.05),
  newtonSteps = 3,
  optN = 20,
  jacobianHScale = 0.5,
  fixRP = c(),
  RecCorrection = "median",
  biasCorrect = FALSE,
  nlminb.control = list(eval.max = 1000, iter.max = 1000),
  ...
)
```

Arguments

`fit` an object to calculate reference points for

`nYears` Number of years to use in per-recruit calculations

`Fsequence` Sequence of F values used to report per-recruit and equilibrium values

`aveYears` Vector of year indices used to calculate average natural mortality, weights, etc. Following ICES guidelines, the default is the last 10 years (starting at 0)

`selYears` Vector of year indices used to calculate selectivity (starting at 0)

`SPRpercent` Vector of x values for F[x * 100%] reference points. Default is 0.35.

`catchType` Catch type used: (total) catch, landings, discard.

`MSYreduction` Vector of proportions for MSY ranges. Default is 0.05 giving an MSY range corresponding to no more than a 5% yield reduction.

`newtonSteps` Number of additional Newton steps at the end of the reference point optimization.

`optN` N used for numerical optimizers to find equilibrium biomass

`jacobianHScale` Scale step size in jacobian calculation

`...` not used

`dYPRpercent` Defunct

`B0percent` Defunct

`fixRP` Defunct

`RecCorrection` Defunct

`biasCorrect` Defunct

`nlminb.control` Defunct

Value

a sam_referencepoints fit

References

Albertsen, C. M. and Trijoulet, V. (2020) Model-based estimates of reference points in an age-based state-space stock assessment model. Fisheries Research, 230, 105618. doi: 10.1016/j.fishres.2020.105618

See Also

forecastMSY

`refit` *Re-fit a model from stockassessment.org*

Description

Re-fit a model from stockassessment.org

Usage

```
refit(fit, newConf, startingValues, ...)
```

Arguments

`fit`	a sam fit or the name of a fit from stockassessment.org
`newConf`	list changes to the configuration
`startingValues`	list of parameter values to use as starting values
`...`	Arguments passed to sam.fit

Value

A new sam fit

`residuals.sam` *Extract residuals from sam object*

Description

Extract residuals from sam object

Usage

```
## S3 method for class 'sam'
residuals(object, discrete = FALSE, ...)
```

Arguments

`object`	sam fitted object as returned from the `sam.fit` function
`discrete`	logical if model contain discrete observations
`...`	extra arguments for TMB's oneStepPredict

Details

one-observation-ahead quantile residuals are calculated

...

retro *retro run*

Description

retro run

Usage

```
retro(fit, year = NULL, ncores = detectCores(), ...)

## S3 method for class 'sam'
retro(fit, year = NULL, ncores = detectCores(), ...)
```

Arguments

`fit`	a fitted model object as returned from sam.fit
`year`	either 1) a single integer n in which case runs where all fleets are reduced by 1, 2, ..., n are returned, 2) a vector of years in which case runs where years from and later are excluded for all fleets, and 3 a matrix of years were each column is a fleet and each column corresponds to a run where the years and later are excluded.
`ncores`	the number of cores to attempt to use
`...`	extra arguments to `sam.fit`

Details

...

rmaxplot *SAM rmax plot*

Description

SAM rmax plot

Usage

```
rmaxplot(fit, ...)

## Default S3 method:
rmaxplot(fit, ...)

## S3 method for class 'samforecast'
rmaxplot(fit, ...)
```

```
## S3 method for class 'hcr'
rmaxplot(fit, ...)
```

Arguments

`fit`	the object returned from sam.fit
`...`	extra arguments transferred to plot including the following: `add` logical, plotting is to be added on existing plot `ci` logical, confidence intervals should be plotted `cicol` color to plot the confidence polygon

Details

Plot of life expectancy

`rmaxtable` *rmax table*

Description

rmax table

Usage

```
rmaxtable(fit, ...)

## Default S3 method:
rmaxtable(fit, ...)
```

Arguments

`fit`	...
`...`	extra arguments not currently used

Details

...

rmvnorm *rmvnorm helper function to draw multivariate normal samples*

Description

rmvnorm helper function to draw multivariate normal samples

Usage

```
rmvnorm(n = 1, mu, Sigma, pivot = FALSE)
```

Arguments

n	the number of samples.
mu	the mean vector.
Sigma	a positive-definite symmetric matrix specifying the covariance matrix.
pivot	Do pivot in chol?

Details

Generates samples via the Cholesky decomposition, which is less platform dependent than eigenvalue decomposition.

Value

If n = 1 a vector of the same length as mu, otherwise an n by length(mu) matrix with one sample in each row.

runwithout *runwithout helper function*

Description

runwithout helper function

Usage

```
runwithout(fit, year, fleet, ...)

## S3 method for class 'sam'
runwithout(fit, year = NULL, fleet = NULL, map = fit$obj$env$map, ...)
```

Arguments

`fit`	a fitted model object as returned from sam.fit
`year`	a vector of years to be excluded. When both fleet and year are supplied they need to be of same length, as only the pairs are excluded
`fleet`	a vector of fleets to be excluded. When both fleet and year are supplied they need to be of same length, as only the pairs are excluded
`...`	extra arguments to sam.fit
`map`	map to use

Details

...

`sam.fit` *Fit SAM model*

Description

Fit SAM model

Usage

```
sam.fit(
  data,
  conf,
  parameters,
  newtonsteps = 3,
  rm.unidentified = FALSE,
  run = TRUE,
  lower = getLowerBounds(parameters, conf),
  upper = getUpperBounds(parameters, conf),
  sim.condRE = TRUE,
  ignore.parm.uncertainty = FALSE,
  rel.tol = 1e-10,
  eval.max = 2000,
  iter.max = 1000,
  penalizeSpline = FALSE,
  fullDerived = FALSE,
  pre.clean = TRUE,
  check.parameters = TRUE,
  ...
)
```

Arguments

`data` data for the sam model as returned from the setup.sam.data function

`conf` model configuration which can be set up using the `defcon` function and then modified either directly in R or by saving it to a text file using the function `saveConf`, modifying the text file, and then reading the configuration from the textfile using the function `loadConf`. For more details about the configuration see details.

`parameters` initial values which can be set up using the `defpar` function and then modified.

`newtonsteps` optional extra true newton steps

`rm.unidentified`
option to eliminate unidentified model parameters based on gradient in initial value (somewhat experimental)

`run` if FALSE return AD object without running the optimization

`lower` named list with lower bounds for optimization (only met before extra newton steps)

`upper` named list with upper bounds for optimization (only met before extra newton steps)

`sim.condRE` logical with default `TRUE`. Simulated observations will be conditional on estimated values of F and N, rather than also simulating F and N forward from their initial values.

`ignore.parm.uncertainty`
option passed to TMB:::sdreport reported uncertainties will not include fixed effect parameter uncertainties

`rel.tol` option passed to stats:::nlminb sets the convergence criteria

`eval.max` option passed to stats:::nlminb sets the maximum number of function evaluations

`iter.max` option passed to stats:::nlminb sets the maximum number of iterations

`penalizeSpline` Add penalization to spline recruitment?

`fullDerived` Report all derived values?

`pre.clean` Should a pre cleaning of data be done?

`check.parameters`
Should parameters be checked in TMB?

`...` extra arguments to MakeADFun

Details

The model configuration object `conf` is a list of different objects defining different parts of the model. The different elements of the list are:

$minAge: A single integer defining the the lowest age class in the assessment.

$maxAge: A single integer defining the the highest age class in the assessment.

$maxAgePlusGroup: Is last age group considered a plus group (1 yes, or 0 no).

$keyLogFsta: A matrix of integers. The number of rows is equal to the number of fleets and the number of columns is equal to the number of age classes. The matrix describes the coupling of the fishing mortality states (the first rows are the catch fleet without effort). '-1' is used for entries where no fishing mortality applies (e.g. age groups in survey fleets, or unobserved age groups). For the valid entries consecutive integers starting at zero must be used, because they are used as indices in the corresponding state vector. If the same number is used for two fleet-age combinations, then the fishing mortality for those are assumed equal (linked to the same state).

$corFlag: An integer vector to specify the correlation structure of log-scale of fishing mortality increments (0 independent, 1 compound symmetry, or 2 AR(1)). The length of the vector is equal to the number of catch fleets without effort information.

$keyLogFpar: A matrix of integers. The number of rows is equal to the number of fleets and the number of columns is equal to the number of age classes. The matrix describes the coupling of survey catchability parameters (so only used for survey fleets). '-1' is used for entries where catchability should not be specified (e.g. fleet - age groups combinations where fishing mortality is specified above, or unobserved fleet - age group combinations). For the valid entries consecutive integers starting at zero must be used, because they are used as indices in the corresponding parameter vector. If the same number is used for two age classes, then the catchability for those age classes are assumed equal (linked to the same parameter).

$keyQpow: A matrix of integers. The number of rows is equal to the number of fleets and the number of columns is equal to the number of age classes. The matrix describes the coupling of density dependent catchability power parameters. This can only be applied to fleets - age combinations where a catchability is defined. '-1' is used for entries where this cannot be applied (e.g. fleet - age groups combinations where fishing mortality is specified above, or unobserved fleet - age group combinations). '-1' is also used to specify that density dependent catchability power parameters is turned off (the most common setup). For entries where density dependent catchability power parameter is to be estimates entries consecutive integers starting at zero must be used. If the same number is used for two age classes, then the density dependent catchability power parameter for those age classes are assumed equal (linked to the same parameter).

$keyVarF: A matrix of integers. The number of rows is equal to the number of fleets and the number of columns is equal to the number of age classes. The matrix describes the coupling of variance parameters for the different states in the log-scale fishing mortality random walk process. '-1' should be used for entries where no fishing mortality state is defined in `keyLogFsta` above. For the valid entries consecutive integers starting at zero must be used, because they are used as indices in the corresponding parameter vector. If the same number is used for two age classes, then the catchability for those age classes are assumed equal (linked to the same parameter). ((a curiosity of this setup is that it is possible to set different variance parameter indices for F-states that are coupled in `keyLogFsta`. This is ignored and the index corresponding to the lowest F-state number is used)).

$keyVarLogN: A vector of integers. The length of the vector is equal to the number of age classes. The vector describes the coupling of variance parameters for the log(N)-process. Consecutive integers starting at zero must be used, because they are used as indices in the corresponding parameter vector. If the same number is used for two age classes, then the catchability for those age classes are assumed equal. A typical setup is to use a unique index for the first age group, because that corresponds to the variance in the (stock-)recruitment, which is often not similar to the variance in the survival process from year to year.

$keyVarObs: A matrix of integers. The number of rows is equal to the number of fleets and the number of columns is equal to the number of age classes. The matrix describes the coupling of observation variance parameters. '-1' should be used for entries where no observations are available. For the valid entries consecutive integers starting at zero must be used, because they are used as indices in the corresponding parameter vector. If the same number is used for two age classes, then the observation variance for those age classes are assumed equal (linked to the same parameter).

$obsCorStruct: A factor specifying the covariance structure used across ages for each fleet. The length of the factor is equal to the number of fleets. The possible options are: ("ID" independent, "AR" AR(1), or "US" for unstructured).

$keyCorObs: A matrix of integers. The number of rows is equal to the number of fleets and the number of columns is equal to the number of age classes _minus_ _one_. The matrix describes the coupling AR correlations between age classes, and hence is only meaningful for fleets where the "AR" observation correlation structure is chosen. '-1' should be used for entries where no observations are available. Notice that the matrix has one column less than the number of age classes, which is because the correlation between age classes is described. Consecutive integers starting at zero must be used. If the same number is used for a given fleet it means that a normal AR(1) structure is used. If different numbers are used for a fleet it means that the correlation parameter changes where the numbers differ. If the "AR" structure is specified above, then the corresponding row in this matrix must have valid non-negative entries.

$stockRecruitmentModelCode: A single integer to specify the stock recruitment connection to use:

Code	**Model**
0	plain random walk on log recruitment
1	Ricker
2	Beverton-Holt
3	piece-wise constant
61	segmented regression (hockey stick)
62	AR(1) on log-recruitment
63	bent hyperbola (smooth hockey stick)
64	power function with degree < 1
65	power function with degree > 1
66	Shepherd
67	Deriso/Hassel
68	Saila-Lorda
69	sigmoidal Beverton-Holt
90	CMP spline (Non-increasing spline on log(R/S))
91	Integrated spline on log(R/S)
92	Spline on log(R/S)

See Albertsen & Trijoulet (2020) for details.

$constRecBreaks: A vector of years to determine piece-wise constant recruitment periods for recruitment model 3. A vector of knot placements on log-SSB for spline recruitment models (90, 91, 92).

$hockeyStickCurve Determines the smoothness of recruitment model 63. The smoothness is estimated if set to NA.

$noScaledYears: A single integer specifying the number of years where catch scaling is to be estimated (most often 0, as this is a somewhat exotic option).

$keyScaledYears: A vector of the years where catch scaling is applied (length should match `noScaledYears`) (most often empty, as this is a somewhat exotic option).

$keyParScaledYA: A matrix of integers specifying the couplings of scale parameters (nrow = `noScaledYears`, ncols = no ages) (most often empty, as this is a somewhat exotic option).

$fbarRange: An integer vector of length 2 specifying lowest and highest age included in Fbar (average fishing mortality summary).

$keyBiomassTreat: A vector of integers with length equal to the number of fleets. '-1' should be used for entries where the corresponding fleet is not a mass index. A the corresponding fleet is a mass index, then three options are available (0 SSB index, 1 catch index, and 2 FSB index).

$obsLikelihoodFlag: A factor specifying the type of likelihood to use for each fleet. The length of the factor is equal to the number of fleets. The possible options are: ("LN" for log-normal and "ALN" Additive logistic normal).

$fixVarToWeight: A single integer. If weight attribute is supplied for observations this option defines how it is treated (0 as relative weight, 1 as a fixed variance = weight).

Value

an object of class `sam`

References

Albertsen, C. M. and Trijoulet, V. (2020) Model-based estimates of reference points in an age-based state-space stock assessment model. Fisheries Research, 230, 105618. doi:10.1016/j.fishres.2020.105618

Examples

```
data(nscodData)
data(nscodConf)
data(nscodParameters)
nscodData$idxCor
storage.mode(nscodData$idxCor)
fit <- sam.fit(nscodData, nscodConf, nscodParameters, silent = TRUE)
```

`saveConf` *Saves a model configuration list to a file*

Description

Saves a model configuration list to a file

Usage

```
saveConf(x, file = "", overwrite = FALSE)
```

Arguments

`x`	sam configuration list as returned from defcon or loadConf
`file`	the file to save the configuration to
`overwrite`	logical if an existing file should be overwritten (FALSE by default)

Details

function useful for saving a model configuration. A saved configuration can be read back in via the loadConf function

`sdplot` *Plots the sd of the log observations as estimated in SAM in increasing order*

Description

Plots the sd of the log observations as estimated in SAM in increasing order

Usage

```
sdplot(fit, barcol = NULL, marg = NULL, ylim = NULL, ...)

## S3 method for class 'sam'
sdplot(fit, barcol = NULL, marg = NULL, ylim = NULL, show.rel.w = FALSE, ...)
```

Arguments

`fit`	the object returned from sam.fit
`barcol`	color for each fleet and age
`marg`	margin for plot (mar in par())
`ylim`	bounds for y-axis
`...`	extra arguments to plot
`show.rel.w`	plots the relative weight of each observation rather than the sd, estimated as (1/sd^2)/max(1/sd^2)

setS *small helper function*

Description

small helper function

Usage

```
setS(x)
```

Arguments

`x`	vector if indices

Details

internal

setSeq *small helper function*

Description

small helper function

Usage

```
setSeq(min, max)
```

Arguments

`min`	from
`max`	to

Details

internal

setup.sam.data *Combine the data sources to SAM readable object*

Description

Combine the data sources to SAM readable object

Usage

```
setup.sam.data(
  fleets = NULL,
  surveys = NULL,
  residual.fleets = NULL,
  prop.mature = NULL,
  stock.mean.weight = NULL,
  catch.mean.weight = NULL,
  dis.mean.weight = NULL,
  land.mean.weight = NULL,
  natural.mortality = NULL,
  prop.f = NULL,
  prop.m = NULL,
  land.frac = NULL,
  recapture = NULL,
  sum.residual.fleets = NULL,
  aux.fleets = NULL,
  keep.all.ages = FALSE,
  average.sampleTimes.survey = TRUE,
  fleetnames.remove.space = TRUE
)
```

Arguments

`fleets` comm fleets vith effort (currently unimplemented)

`surveys` surveys

`residual.fleets` fleet, or list of fleets without effort information

`prop.mature` pm

`stock.mean.weight` sw

`catch.mean.weight` cw

`dis.mean.weight` dw

`land.mean.weight` lw

`natural.mortality`
: nm

`prop.f`
: ...

`prop.m`
: ...

`land.frac`
: ...

`recapture`
: ...

`sum.residual.fleets`
: ...

`aux.fleets`
: ...

`keep.all.ages`
: ...

`average.sampleTimes.survey`
: Should sample times for surveys be averaged?

`fleetnames.remove.space`
: Should white space in fleet names be removed?

Details

...

---

simstudy *Simulate data from fitted model and re-estimate from each run*

Description

Simulate data from fitted model and re-estimate from each run

Usage

```
simstudy(fit, nsim, ncores = detectCores())
```

Arguments

`fit`	a fitted model returned from sam.fit
`nsim`	number of simulations
`ncores`	number of cores to be used

simulate.sam *Simulate from a sam object*

Description

Simulate from a sam object

Usage

```
## S3 method for class 'sam'
simulate(
  object,
  nsim = 1,
  seed = NULL,
  full.data = TRUE,
  keep.process = FALSE,
  retain.missing = FALSE,
  ...
)
```

Arguments

`object`	sam fitted object as returned from the `sam.fit` function
`nsim`	number of response lists to simulate. Defaults to 1.
`seed`	random number seed
`full.data`	logical, should each inner list contain a full list of data. Defaults to TRUE
`keep.process`	Keep logN and logF processes when full.data = TRUE?
`retain.missing`	Keep NA in places where observations are missing?
`...`	extra arguments

Details

simulates data sets from the model fitted and conditioned on the random effects estimated

Value

returns a list of lists. The outer list has length `nsim`. Each inner list contains simulated values of `logF`, `logN`, and obs with dimensions equal to those parameters.

`sprplot` *SAM SPR plot*

Description

SAM SPR plot

Usage

```
sprplot(fit, ...)

## Default S3 method:
sprplot(fit, ...)

## S3 method for class 'samforecast'
sprplot(fit, ...)

## S3 method for class 'hcr'
sprplot(fit, ...)
```

Arguments

`fit`	the object returned from sam.fit
`...`	extra arguments transferred to plot including the following: `add` logical, plotting is to be added on existing plot `ci` logical, confidence intervals should be plotted `cicol` color to plot the confidence polygon

Details

Plot of deterministic equilibrium spawners per recruit assuming biological parameters and selectivity for that year remains unchanged in the future.

`sprtable` *SPR table*

Description

SPR table

Usage

```
sprtable(fit, ...)

## Default S3 method:
sprtable(fit, ...)
```

Arguments

`fit`	...
`...`	extra arguments not currently used

Details

...

`srplot` *Plots the stock recruitment*

Description

Plots the stock recruitment

Usage

```
srplot(fit, ...)

## S3 method for class 'sam'
srplot(
  fit,
  textcol = "red",
  years = TRUE,
  linetype = "l",
  linecol = "black",
  polycol = do.call("rgb", c(as.list(col2rgb("black")[, 1]), list(alpha = 0.1))),
  polyborder = do.call("rgb", c(as.list(col2rgb("black")[, 1]), list(alpha = 0.3))),
  polylty = 3,
  polylwd = 1,
  xlim,
  ylim,
  add = FALSE,
  CIlevel = 0.95,
  addCurve = TRUE,
  ...
)
```

Arguments

`fit`	the object returned from sam.fit
`...`	extra arguments to plot
`textcol`	color of years on plot
`years`	the plotting symbols are the years
`linetype`	type for the plot (default line)

`linecol`	color of lines between points
`polycol`	Inner color of error ellipses
`polyborder`	Border color of error ellipses
`polylty`	Border line type of error ellipses
`polylwd`	Border line width of error ellipses
`xlim`	bounds for x-axis
`ylim`	bounds for y-axis
`add`	false if a new plot should be created
`CIlevel`	Confidence level for error ellipses on stock-recruitment pairs
`addCurve`	Call addRecruitmentCurve?

`ssbplot` *SAM SSB plot*

Description

SAM SSB plot

Usage

```
ssbplot(fit, ...)

## Default S3 method:
ssbplot(fit, ...)

## S3 method for class 'samforecast'
ssbplot(fit, ...)

## S3 method for class 'hcr'
ssbplot(fit, ...)
```

Arguments

`fit`	the object returned from sam.fit
`...`	extra arguments transferred to plot including the following: `add` logical, plotting is to be added on existing plot `ci` logical, confidence intervals should be plotted `cicol` color to plot the confidence polygon

Details

Plot of spawning stock biomass

ssbtable *SSB table*

Description

SSB table

Usage

```
ssbtable(fit, ...)

## Default S3 method:
ssbtable(fit, ...)
```

Arguments

`fit`	...
`...`	extra arguments not currently used

Details

...

stochasticReferencepoints *Estimate stochastic reference points*

Description

The function estimates reference points based on stochastic model forecasts.

Usage

```
stochasticReferencepoints(fit, referencepoints, ...)

## S3 method for class 'sam'
stochasticReferencepoints(
  fit,
  referencepoints,
  method = "Median",
  catchType = "catch",
  nYears = 100,
  Frange = c(0, 2),
  nosim = 200,
  aveYears = max(fit$data$years) + (-9:0),
  selYears = max(fit$data$years),
  newton.control = list(),
  seed = .timeToSeed(),
  knots = round(nosim/20),
  nosim_ci = 200,
  derivedSummarizer = NA,
  nTail = 1,
  constraint = "F=%f",
  deterministicF = TRUE,
  Fsequence = seq(Frange[1], Frange[2], len = 50),
  run = TRUE,
  DT = 0,
  equilibriumMethod = c("EC", "ES", "AD"),
  ncores = 1,
  ...
)
```

Arguments

`fit`	a sam fit
`referencepoints`	a character vector of reference points to estimate (see Details)
`...`	additional parameters that can be passed on
`method`	estimation method (See Details)
`catchType`	catch type: catch, landing, discard
`nYears`	Number of years to forecast
`Frange`	Range of F values to consider
`nosim`	Number of simulations for estimation
`aveYears`	Years to average over for biological input
`selYears`	Years to average over for selectivity
`newton.control`	List of control parameters for optimization
`seed`	Seed for simulations
`knots`	Number of knots to use
`nosim_ci`	Number of simulations for bootstrap confidence intervals
`derivedSummarizer`	Function to summarize derived per-recruit values
`nTail`	Number of years from the simulation to include in calculations
`constraint`	Format of forecast constraint. "%f" is replaced by F values.
`deterministicF`	If FALSE, modelled logF process noise will be added to target logF in forecasts.
`Fsequence`	F sequence to explore
`run`	run it?
`DT`	...
`equilibriumMethod`	method to use
`ncores`	Number of cores

Details

The following reference points are implemented:

F=x F fixed to x, e.g., "F=0.3" (NOT IMPLEMENTED YET)

StatusQuo F in the last year of the assessment (NOT IMPLEMENTED YET)

StatusQuo-y F in the y years before the last in the assessment, e.g., "StatusQuo-1" (NOT IMPLEMENTED YET)

MSY F that maximizes yield

0.xMSY Fs that gives 0.x*100% of MSY, e.g., "0.95MSY"

Max F that maximizes yield per recruit (NOT IMPLEMENTED YET)

0.xdYPR F such that the derivative of yield per recruit is 0.x times the derivative at F=0, e.g., "0.1dYPR" (NOT IMPLEMENTED YET)

0.xSPR F such that spawners per recruit is 0.x times spawners per recruit at F=0, e.g., "0.35SPR" (NOT IMPLEMENTED YET)

0.xB0 F such that biomass is 0.x times the biomass at F=0, e.g., "0.2B0" (NOT IMPLEMENTED YET)

Reference points can be estimated using these methods:

Mean Use least squares to estimate mean equilibrium values

Q0.x Use quantile regression to estimate the 0.x quantile of equilibrium values

Median Identical to Q0.5

Mode (NOT IMPLEMENTED YET)

To estimate median equilibrium yield, as required by ICES, the method "Q0.5" should be used. Note that this function is highly experimental.

Value

reference point object

Examples

```
## Not run:
  stochasticReferencepoints(fit, c("MSY","0.95MSY"))

## End(Not run)
```

stockassessment-deprecated *Deprecated and defunct functions*

Description

Deprecated and defunct functions

referencepoints

For `referencepoints`, use `deterministicReferencepoints`.

summary.sam *Summary of sam object*

Description

Summary of sam object

Usage

```
## S3 method for class 'sam'
summary(object, ...)
```

Arguments

`object`	sam fitted object as returned from the `sam.fit` function
`...`	extra arguments

Details

summary table containing recruits, SSB, and Fbar

tableit *Table helper*

Description

Table helper

Usage

```
tableit(fit, what, x = fit$data$years, trans = function(x) x, ...)

## S3 method for class 'sam'
tableit(fit, what, x = fit$data$years, trans = function(x) x, ...)

## S3 method for class 'samforecast'
tableit(fit, what, x = fit$data$years, trans = function(x) x, ...)
```

Arguments

fit	returned object from sam.fit
what	quoted name of what to extract
x	rownames of table
trans	function to be applied
...	extra arguments not currently used

Details

...

tsbplot *SAM TSB plot*

Description

SAM TSB plot

Usage

```
tsbplot(fit, ...)

## Default S3 method:
tsbplot(fit, ...)
```

Arguments

`fit` the object returned from sam.fit

`...` extra arguments transferred to plot including the following:
`add` logical, plotting is to be added on existing plot
`ci` logical, confidence intervals should be plotted
`cicol` color to plot the confidence polygon

Details

Plot of total stock biomass

`tsbtable` *TSB table*

Description

TSB table

Usage

```
tsbtable(fit, ...)

## Default S3 method:
tsbtable(fit, ...)
```

Arguments

`fit` ...

`...` extra arguments not currently used

Details

...

`write.data.files` *Write all data files from a list as created by 'setup.sam.data'*

Description

Write all data files from a list as created by 'setup.sam.data'

Usage

```
write.data.files(dat, dir = ".", writeToOne = TRUE, ...)
```

Arguments

`dat`	A list as created by 'setup.sam.data'
`dir`	Directory where the files are written
`writeToOne`	Write multi fleet data to one file if data is equal for all fleets
`...`	other arguments passes to write.ices

Details

Write all data files from a list as created by 'setup.sam.data'

`write.ices` *Write ICES/CEFAS data file from matrix*

Description

Write ICES/CEFAS data file from matrix

Usage

```
write.ices(x, fileout, writeToOne = TRUE, ...)
```

Arguments

`x`	a matrix where rownames are taken as years and colnames are taken as ages
`fileout`	file name or connection
`writeToOne`	Write multi fleet data to one file if data is equal for all fleets
`...`	Arguments to be passed to write

Details

Takes the data and writes them in the ICES/CEFAS format. It is assumed that rows represent consecutive years and cols consecutive ages

`write.surveys` *Write surveys in ICES/CEFAS data file from a model object*

Description

Write surveys in ICES/CEFAS data file from a model object

Usage

```
write.surveys(fit, fileout, ...)
```

Arguments

`fit`	A fitted object as returned from sam.fit
`fileout`	file name or connection
`...`	Arguments to be passed to write

Details

Takes the survey data from the fitted object and writes them in the ICES/CEFAS format.

`yearslostplot` *SAM years lost to fishing plot*

Description

SAM years lost to fishing plot

Usage

```
yearslostplot(fit, cause, ...)

## Default S3 method:
yearslostplot(fit, cause = c("Fishing", "Other", "LifeExpectancy"), ...)

## S3 method for class 'samforecast'
yearslostplot(fit, cause = c("Fishing", "Other", "LifeExpectancy"), ...)

## S3 method for class 'hcr'
yearslostplot(fit, cause = c("Fishing", "Other", "LifeExpectancy"), ...)
```

Arguments

`fit` the object returned from sam.fit

`cause` Fisning, Other, or LifeExpectancy

`...` extra arguments transferred to plot including the following:
`add` logical, plotting is to be added on existing plot
`ci` logical, confidence intervals should be plotted
`cicol` color to plot the confidence polygon

Details

Plot of years lost to fishing

`yearslosttable` *Years Lost table*

Description

Years Lost table

Usage

```
yearslosttable(fit, cause, ...)

## Default S3 method:
yearslosttable(fit, cause = c("Fishing", "Other", "LifeExpectancy"), ...)
```

Arguments

`fit` ...

`cause` Fisning, Other, or LifeExpectancy

`...` extra arguments not currently used

Details

...

ypr *Yield per recruit calculation*

Description

Yield per recruit calculation

Usage

```
ypr(
  fit,
  Flimit = 2,
  Fdelta = 0.01,
  aveYears = min(15, length(fit$data$years)),
  ageLimit = 100,
  sprProp = 0.35,
  ...
)

## S3 method for class 'sam'
ypr(
  fit,
  Flimit = 2,
  Fdelta = 0.01,
  aveYears = min(15, length(fit$data$years)),
  ageLimit = 100,
  sprProp = 0.35,
  ...
)
```

Arguments

`fit`	the object returned from sam.fit
`Flimit`	Upper limit for Fbar
`Fdelta`	increments on the Fbar axis
`aveYears`	Number of years back to use when calculating averages (selection, weights, ...)
`ageLimit`	Oldest age used (should be high)
`sprProp`	Proportion of SPR at F=0, for example 0.35 if F0.35SPR
`...`	extra arguments not currently used

`yprplot` *SAM YPR plot*

Description

SAM YPR plot

Usage

```
yprplot(fit, ...)

## Default S3 method:
yprplot(fit, ...)

## S3 method for class 'samforecast'
yprplot(fit, ...)

## S3 method for class 'hcr'
yprplot(fit, ...)
```

Arguments

`fit`	the object returned from sam.fit
`...`	extra arguments transferred to plot including the following: `add` logical, plotting is to be added on existing plot `ci` logical, confidence intervals should be plotted `cicol` color to plot the confidence polygon

Details

Plot of deterministic equilibrium yield per recruit assuming biological parameters and selectivity for that year remains unchanged in the future.

`yprtable` *YPR table*

Description

YPR table

Usage

```
yprtable(fit, ...)

## Default S3 method:
yprtable(fit, ...)
```

Arguments

`fit` ...

`...` extra arguments not currently used

Details

...

Index

∗ **datasets**
nscodConf, 54
nscodData, 55
nscodParameters, 55
.SAM_replicate, 4

addforecast, 5
addRecruitmentCurve, 6

b0plot, 7
b0table, 8
bc, 9

c.sam, 9
catchbyfleetplot, 10
catchbyfleettable, 10
catchplot, 11
catchtable, 12
caytable, 12
clean.void.catches, 13
coef.sam, 13, *68*
componentplot, 14
corplot, 15
corplotcommon, 15

dataplot, 16
defcon, *13*, 16, *84*
defpar, 17, *84*
deterministicReferencepoints, 18, *99*

empirobscorrplot, 19
equilibriumbiomassplot, 20
equilibriumbiomasstable, 21
erbplot, 21
erbtable, 22

faytable, 22
fbarplot, 23
fbartable, 25
fitfromweb, 25
fitplot, 26
forecast, 26, *29*, *49*, *50*
forecastMSY, 28, *78*
fselectivityplot, 30

generationlengthplot, 30
generationlengthtable, 31
getAllDerivedValues, 31
getFleet, 32
getLowerBounds, 32
getResidualFleets, 33
getUpperBounds, 33
grad, 34

hcr, 34, *38*, *59*, *67*
hessian, 36

ibc, 37
icesAdviceRule, 37
iibc, 38
is.whole.positive.number, 39

jacobian, 40
jit, 40

leaveout, 41
lifeexpectancyplot, 42
lifeexpectancytable, 43
loadConf, 43, *84*
logLik.sam, 44

modelDescription, 44
modelforecast, *35*, 45
modeltable, 51
modelVersionInfo, 51
mohn, 52
MSE, 52

newton, *19*
nobs.sam, 53
nscodConf, 54
nscodData, 55

nscodParameters, 55
ntable, 56

obscorrplot, 57
obscov, 57

parplot, 58
partable, 58
plot.hcr, 59
plot.sam, 59
plot.samforecast, 60
plot.samres, 60
plot.samset, 61
plot.samypr, 61
plotby, 62
plotit, 63
predstdplot, 66
print.hcr, 67
print.sam, 67
print.sam_referencepoints, 70
print.samcoef, 68
print.samforecast, 68
print.samres, 69
print.samset, 69
print.samypr, 70
procres, *60*, *69*, 71

qtable, 71
qtable.sam, 72
qtableplot, 72

read.data.files, 73
read.ices, 73
read.surveys, 74
read.table.nowarn, 74
recplot, *59*, 75
rectable, 75
reduce, 76
referencepoints, *29*, *70*, 77
refit, 79
residuals.sam, *60*, *69*, 79
retro, 80
rmaxplot, 80
rmaxtable, 81
rmvnorm, 82
runwithout, 82

sam.fit, *9*, *13*, *42*, *44*, *53*, *59–61*, *67*, *71*, *79*, *80*, 83, *92*, *99*
saveConf, *84*, 87
sdplot, 88
sdreport, *19*
setS, 89
setSeq, 89
setup.sam.data, 90
simstudy, 91
simulate.sam, 92
sprplot, 93
sprtable, 93
srplot, 94
ssbplot, *59*, 95
ssbtable, 96
stochasticReferencepoints, 96
stockassessment-deprecated, 99
summary.sam, 99

tableit, 100
tsbplot, 100
tsbtable, 101

write.data.files, 101
write.ices, 102
write.surveys, 103

yearslostplot, 103
yearslosttable, 104
ypr, 105
yprplot, 106
yprtable, 106