Package: spict (via r-universe)

May 25, 2026

Type Package

Title Stochastic surplus Production model in Continuous-Time (SPiCT)

Version 1.3.9

Date 2026-04-17

Description Fits a surplus production model to fisheries catch and biomass index data.

License GPL (>=3)

Depends R (>= 3.0), TMB (>= 1.7.1)

LinkingTo TMB, RcppEigen

Imports ellipse

Suggests parallel, mgcv, rjags, coda, knitr, rmarkdown

LazyData true

Encoding UTF-8

VignetteBuilder knitr

RoxygenNote 7.3.3

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

Date/Publication 2026-05-25 08:04:10 UTC

RemoteUrl https://github.com/iagomosqueira/spict

RemoteRef HEAD

RemoteSha 39595209f1ee0c52662b55f6058f4de102afe8e3

RemoteSubdir spict

Contents

acf.signf *Check whether ACF of residuals is significant in any lags.*

Description

Check whether ACF of residuals is significant in any lags.

Usage

```
acf.signf(resid, lag.max = 4, return.p = FALSE)
```

Arguments

resid	Vector of residuals.
lag.max	Only check from lag 1 until lag.max.
return.p	Return p-values of the calculated lags.

Details

This corresponds to plotting the ACF using acf() and checking whether any lags has an acf value above the CI limit.

Value

Vector of TRUE and FALSE indicating whether significant lags were present. If return.p is TRUE then p-values are returned instead.

add.catchunit *Add catch unit to label*

Description

Add catch unit to label

Usage

```
add.catchunit(lab, cu)
```

Arguments

lab	Base label
cu	Catch unit as a character string

Value

Label with added catch unit

add.col.legend *Add a legend explaining colors of points (vertical orientation)*

Description

Add a legend explaining colors of points (vertical orientation)

Usage

```
add.col.legend()
```

Value

Nothing.

add.col.legend.hor *Add a legend explaining colors of points (horizontal orientation)*

Description

Add a legend explaining colors of points (horizontal orientation)

Usage

```
add.col.legend.hor()
```

Value

Nothing.

add.man.scenario *Define management scenario*

Description

Define management scenario

Usage

```
add.man.scenario(
  rep,
  scenarioTitle = "",
  maninterval = NULL,
  maneval = NULL,
  ffac = NULL,
  fabs = NULL,
  cfac = NULL,
  cabs = NULL,
  fractiles = list(catch = 0.5, bbmsy = 0.5, ffmsy = 0.5),
  breakpointB = 0,
  limitB = 0,
  safeguardB = list(limitB = 0, prob = 0.95),
  intermediatePeriodCatch = NULL,
  intermediatePeriodCatchSDFac = 1,
  intermediatePeriodCatchList = NULL,
  ctol = 0.001,
  evalBreakpointB = 0,
  evalLimitB = 0,
  verbose = TRUE,
  dbg = 0,
  mancheck = TRUE
)

get.TAC(
  rep,
  scenarioTitle = "",
  maninterval = NULL,
  maneval = NULL,
  ffac = NULL,
  fabs = NULL,
  cfac = NULL,
  cabs = NULL,
  fractiles = list(catch = 0.5, bbmsy = 0.5, ffmsy = 0.5),
  breakpointB = 0,
  limitB = 0,
  safeguardB = list(limitB = 0, prob = 0.95),
  intermediatePeriodCatch = NULL,
  intermediatePeriodCatchSDFac = 1,
  intermediatePeriodCatchList = NULL,
  ctol = 0.001,
  evalBreakpointB = 0,
  evalLimitB = 0,
  verbose = TRUE,
  dbg = 0,
  mancheck = TRUE
)
```

```
make.man.inp(
  rep,
  scenarioTitle = "",
  maninterval = NULL,
  maneval = NULL,
  ffac = NULL,
  fabs = NULL,
  cfac = NULL,
  cabs = NULL,
  fractiles = list(catch = 0.5, bbmsy = 0.5, ffmsy = 0.5),
  breakpointB = 0,
  limitB = 0,
  safeguardB = list(limitB = 0, prob = 0.95),
  intermediatePeriodCatch = NULL,
  intermediatePeriodCatchSDFac = 1,
  intermediatePeriodCatchList = NULL,
  ctol = 0.001,
  evalBreakpointB = 0,
  evalLimitB = 0,
  verbose = TRUE,
  dbg = 0,
  mancheck = TRUE
)
```

Arguments

rep	A result report as generated by running `fit.spict`.
scenarioTitle	Title of scenario (default: `'customScenario_X'`, where X is an integer equal to the number of scenarios with the same name in `rep$man` plus 1, e.g. `'customScenario_3'`).
maninterval	Two floats representing the start and end of the management period. Example: `maninterval = c(2020.25,2021.25)`. Default: NULL.
maneval	Time at which to evaluate model states. Example: `maneval= 2021.25`. Default: NULL.
ffac	Factor to multiply current fishing mortality by (default: NULL).
fabs	Absolute fishing mortality for management period (default: NULL).
cfac	Factor to multiply current catch by (default: NULL). Please refer to the details for more information.
cabs	Absolute catch for the management period (default: NULL).
fractiles	List defining the fractiles of the 3 distributions of 'catch', 'bbmsy', and 'ffmsy'. By default (0.5) median is used for all 3 quantities. Please refer to the details for more information.
breakpointB	Breakpoints in terms of B/B_{MSY} for the hockey-stick HCR. By default (0) no breakpoint is assumed. If one value is provided, F is reduced linearly to zero, if B/B_{MSY} is below the breakpoint. If two values ara provided, F is reduced linearly to the lower of the two provided values, if B/B_{MSY} is below the higher

and above the lower value, and F is zero if B/B_{MSY} is below the lower value. The higher value corresponds to a biomass threshold where F is reduced and the lower to a limit where F=0. Note that the breakpoints are evaluated at the start of the management period. Please refer to the details for more information.

`limitB` Biomass limit as relative to B_MSY, i.e. B/B_{MSY}. By default (0) no limit is assumed. If B/B_{MSY} is below the limitB (e.g. 0.3), F is set to 0. Note that the limit is evaluated at the start of the management period.

`safeguardB` List defining an optional precautionary buffer by means of a biomass reference level relative to B/B_{MSY} (`'limitB'`; default: 0, i.e. deactivating the PA buffer) and the risk aversion probability (`'prob'`; default: 0.95). Please refer to the details for more information.

`intermediatePeriodCatch`
Catch during intermediate period, e.g. last year's TAC (default: `NULL`). Please refer to the details for more information.

`intermediatePeriodCatchSDFac`
Factor for the multiplication of the standard deviation of the catch during the intermediate period (default: 1). Please refer to the details for more information.

`intermediatePeriodCatchList`
List defining catch in the intermediate period obtaining the elements 'obsC', 'timeC', and 'dtc' (optional element 'stdevfacC' which is 1 if not provided). Please refer to the details for more information.

`ctol` Tolerance of `nlminb` when finding F that leads to provided target catch (via arguments `cfac` or `cabs`)

`evalBreakpointB`
Time for the evaluation of the hockey-stick component of the HCR: 0 indicating start of the management period and 1 indicating the end of the management period (default: 0).

`evalLimitB` Time for the evaluation of the biomass limit component of the HCR: 0 indicating start of the management period and 1 indicating the end of the management period (default: 0).

`verbose` Should detailed outputs be provided (default: TRUE).

`dbg` Debug flag, dbg=1 some output, dbg=2 more output.

`mancheck` Should the time-dependent objects in `inp` be checked against the management time and corrected if necessary? (Default: TRUE)

Details

Default management scenario: The default management scenario is fish at F_{MSY}. This is when `ffac`, `cfac`, `fabs`, `cabs` are all `NULL`, and `breakpointB` and `safeguardB$limitB` are 0. In practice `ffac` is set equal to F_{MSY}/F_m.

Catch scenarios: Management scenarios can be defined based on a desired catch during the management period. Common examples include scenarios like "increase catch by 25%", "keep current catch", or "zero catch". The catch can be relative to the predicted "previous catch", using the multiplier `cfac`, or in absolute terms using `cabs` catch in the same units as the input data. By default, the respective previous catch corresponds to that part of the previous year which corresponds to the management interval. For example, if the management period is [1991, 1992),

the whole catch from the year [1990, 1991) is being used. If the management period is [1991.5, 1991.75), the same interval from the previous year [1990.5, 1990.75) is being used. If the management period spans several years, e.g. [1991, 1993), the whole catch from the previous year [1990, 1991) is being used two times.

Harvest Control Rules (HCRs):

The combination of the arguments "fractiles", "breakpointB", and "safeguardB" allow the specification of a number of different harvest control rules:

- MSY hockey-stick rule: Fishing at F_MSY above a certain biomass reference level (here defined as a fraction of B_MSY with `breakpointB`). Below the reference level, fishing is reduced linearly to 0 as suggested in ICES (2017).
- MSY (hockey-stick) rule with additional precautionary buffer: As long as the probability of the predicted biomass relative to a reference biomass level (e.g. 0.3 B_MSY, defined by `safeguardB$limitB`) is smaller or equal to a specified risk aversion probability (e.g. 95%, defined by `safeguardB$prob`), fishing at F_MSY or following the hockey-stick rule (if `breakpointB != 0`), otherwise reduce fishing mortality to meet specified risk aversion probability (`safeguardB$prob`) as introduced in ICES (2018).
- A MSY hockey-stick rule that accounts of uncertainty of estimated quantities: Fishing at 35th percentile of F_MSY above the 35th percentile of 0.5 B/B_{MSY} (`breakpointB = 0.5`) and 35th percentile of linearly reduced F_MSY below the 35th percentile of 0.5 B/B_{MSY}. TAC corresponds to 35th percentile of predicted catch. Rule is applied with `fractiles = list(catch=0.35, bbmsy=0.35, ffmsy=0.35)`, `breakpointB = 0.5` as introduced in ICES (2019).

Fractiles: By default, the median (fractile of 0.5) is used for the stock status (B/B_{MSY}, F/F_{MSY}) and predicted catch distribution. A more precautionary approach is to used fractiles lower than the median (0.5) to account for the estimated uncertainty. The arguments of the 'fractiles' are:

- catch - Fractile of the predicted catch distribution
- bbmsy - Fractile of the B/B_{MSY} distribution
- ffmsy - Fractile of the F/F_{MSY} distribution

Note that the fractile for the F/F_{MSY} distribution is 1 minus the fractile specified. As the current fishing mortality is divided by the value of this distribution $F_{y+1} = \frac{F_y}{F_y/F_{MSY}}$, a lower percentile of the F/F_{MSY} distribution is more conservative than a larger one. This allows a consistent setting of fractiles among the different quantities.

Biomass safeguard: The argument list "safeguardB" includes:

- limitB - Reference level for the evaluation of the predicted biomass defined as fraction of B/B_{MSY}. By default (`safeguardB$limitB == 0`) the PA buffer is not used. Theoretically, any value smaller than 1 is meaningful, but an ICES recommended value would be 30% `safeguardB$limitB = 0.3` (ICES, 2018).
- prob - Risk aversion probability of the predicted biomass relative to specified reference level (`safeguardB$limitB`) for all rules with PA buffer (`safeguardB$limitB != 0`). Default: 0.95 as recommended by ICES (2018).

Intermediate period assumptions: Dependent on the start of the management period (e.g. advice year), there might be a time lag between the last observation and the start of the management

period, often referred to as the intermediate period. If this is the case, an assumption about the catch during intermediate time period (e.g. assessment year) has to be made. Two meaningful assumptions are:

1: The catch in the intermediate period is based on the fishing mortality which is extrapolated from the previous year. This is the default assumption;

2: The catch in the intermediate period is directly specified. This could for example be the TAC recommended in the previous year. The catch can be specified by means of the argument `intermediatePeriodCatch`. Be aware that this catch might correspond to several years or a fraction of a year depending on the time between the last observation and the start of the management period. The function `man.timeline` can help visualising the default or specified intermediate period in your data. The argument `intermediatePeriodCatchSDFac` allows to specify the factor with which to multiply the standard deviation of the catch (σ_C) with. It is thus a measure of the certainty around the catch in the intermediate period. The argument `intermediatePeriodCatchList` allows to define a list with catches and their intervals. It is a list with the elements 'obsC', 'timeC', 'dtc' and the optional element 'stdevfacC' (which is equal to 1 if not provided).

`make.man.inp`**:** Internal function that creates the required input list for the specific HCR.

Value

`add.man.scenario` returns the input object `rep` with the specified HCR added to the `man` list. `get.TAC` returns the total allowable catch (TAC) based on the specified scenario. `make.man.inp` returns the updated `inp` list based on specified HCR.

References

ICES. 2017. Report of the Workshop on the Development of the ICES approach to providing MSY advice for category 3 and 4 stocks (WKMSYCat34), 6-10 March 2017, Copenhagen, Denmark. ICES CM 2017/ ACOM:47. 53 pp.

ICES. 2018. Report of the Eighth Workshop on the Development of Quantitative Assessment Methodologies based on LIFE-history traits, exploitation characteristics, and other relevant parameters for data-limited stocks (WKLIFE VIII), 8-12 October 2018, Lisbon, Portugal. ICES CM 2018/ACOM:40. 172 pp.

ICES.2019. Ninth Workshop on the Development of Quantitative Assessment Methodologies based on LIFE-history traits, exploitation characteristics, and other relevant parameters for data-limited stocks (WKLIFE IX). ICES Scientific Reports. 1:77. 131 pp.http://doi.org/10.17895/ices.pub.5550

ICES 2020. Report of the Ninth Workshop on the Development of Quantitative Assessment Methodologies based on LIFE-history traits, exploitation characteristics, and other relevant parameters for data-limited stocks (WKLIFE X), ICES Scientific Reports. 2:98. 72 pp. http://doi.org/10.17895/ices.pub.5985

Examples

```
data(pol)
rep <- fit.spict(pol$albacore)

## Fishing at Fmsy
rep <- add.man.scenario(rep)
```

```
## MSY hockey-stick rule
rep <- add.man.scenario(rep, breakpointB = 0.5)

## MSY hockey-stick rule accounting for uncertainty
rep <- add.man.scenario(rep, fractiles = list(catch=0.35, bbmsy=0.35, ffmsy=0.35), breakpointB=0.5)

## MSY hockey-stick rule with linearly decreasing F between 0.3 and 0.5 Bmsy
rep <- add.man.scenario(rep, fractiles = list(catch=0.35), breakpointB = c(0.3, 0.5))

## Now `rep` includes 4 management scenarios

## Get the TAC when fishing mortality is equal to Fmsy
get.TAC(rep)

## Get TAC for the MSY hockey-stick rule (only using Btrigger)
get.TAC(rep, breakpointB = 0.5)

## Get TAC for the MSY hockey-stick rule (with Btrigger and Blim)
get.TAC(rep, breakpointB = c(0.3, 0.5))

## Get the TAC for the HCR accounting for uncertainty
get.TAC(rep, fractiles = list(catch=0.35, bbmsy=0.35, ffmsy=0.35), breakpointB=0.5)

## Get the TAC for the ICES (2025) recommended HCR (as used in WKMSYSPICT)
get.TAC(rep, fractiles = list(catch=0.35), breakpointB = 0.5, limitB = 0.3)
```

`add.manlines` *Add lines to plot indicating result of management scenarios.*

Description

Add lines to plot indicating result of management scenarios.

Usage

```
add.manlines(
  rep,
  par,
  par2 = NULL,
  index.shift = 0,
  plot.legend = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

rep	A result report as generated by running fit.spict.
par	The name of the parameter to be plotted.
par2	If a second parameter should be used as explanatory variable instead of time.
index.shift	Shift initial time point by this index.
plot.legend	Logical; should the legend be plotted?
verbose	Should detailed outputs be provided (default: TRUE).
...	Passed to lines.

Value

Nothing

---

annual *Convert from quarterly (or other sub-annual) data to annual means, sums or a custom function.*

Description

Convert from quarterly (or other sub-annual) data to annual means, sums or a custom function.

Usage

```
annual(intime, vec, type = "mean")
```

Arguments

intime	A time vector corresponding to the values in vec.
vec	The vector of values to convert to annual means.
type	item to match as function: symbol or string, see match.fun for details.

Value

A list containing the annual means $annvec and a corresponding time vector $anntime.

`arrow.line` *Draw a line with arrow heads.*

Description

Draw a line with arrow heads.

Usage

```
arrow.line(
  x,
  y,
  length = 0.25,
  angle = 30,
  code = 2,
  col = par("fg"),
  lty = par("lty"),
  lwd = par("lwd"),
  ...
)
```

Arguments

`x`	X coordinates.
`y`	Y coordinates.
`length`	See documentation for arrows.
`angle`	See documentation for arrows.
`code`	See documentation for arrows.
`col`	See documentation for arrows.
`lty`	See documentation for arrows.
`lwd`	See documentation for arrows.
`...`	See documentation for arrows.

Details

Add to an existing plot a continuous line with arrow heads showing the direction between each data point

Value

Nothing, but an arrow line is added to the current plot.

calc.bmsyk *Calculates the Bmsy/K ratio*

Description

Calculates the Bmsy/K ratio

Usage

```
calc.bmsyk(rep)
```

Arguments

rep	Result of fit.spict().

Value

Bmsy/K

calc.EBinf *Calculate E(Binfinity), i.e. the fished equilibrium.*

Description

Calculate E(Binfinity), i.e. the fished equilibrium.

Usage

```
calc.EBinf(K, n, Fl, Fmsy, sdb2)
```

Arguments

K	The carrying capacity.
n	Pella-Tomlinson exponent.
Fl	Average fishing mortality of the last year.
Fmsy	Fishing mortality at MSY.
sdb2	Standard deviation squared (variance) of B process.

Details

If a seasonal pattern in F is imposed the annual average F is used for calculating the expectation. Max() is used to avoid negative values.

Value

E(Binf).

calc.gamma *Calculate gamma from n*

Description

Calculate gamma from n

Usage

```
calc.gamma(n)
```

Arguments

n	Exponent of the Pella-Tomlinson surplus production equation.

calc.influence *Calculates influence statistics of observations.*

Description

Calculates influence statistics of observations.

Usage

```
calc.influence(rep, mc.cores = 1)
```

Arguments

rep	A valid result from fit.spict().
mc.cores	Number of cores for `parallel::mclapply` function. By default 1.

Details

TBA

Value

A list equal to the input with the added key "infl" containing influence statistics.

`calc.mase` *Calculate Mean Absolute Scaled Error (MASE)*

Description

Calculate Mean Absolute Scaled Error (MASE)

Usage

```
calc.mase(rep, verbose = TRUE)
```

Arguments

`rep`	Result of `fit.spict` that contains hindcasted runs added by `hindcast`.
`verbose`	Should detailed outputs be provided (default: TRUE).

Details

This function calculates the Mean Absolute Scaled Error (MASE) for each index time series by hindcasting. Thus, the application of this method requires a fitted spict object with the results of the hindcasting analysis `hindcast`.

The smaller MASE, the higher the predictive power of the spict model regarding the prediction of index observations. In contrast, a MASE above 1 suggests that the naive prediction of the index observations assuming the preceding index observations have a higher predictive power than the spict model. Note, however, that the absolute MASE value depends on multiple factors such as the number of peels, assumed priors, etc.

Note that a difference in the timing of the index observations of less than a month are considered acceptable for the estimation of the naive prediction residuals and no warning is printed. If the variability exceeds a month, the predictions are still calculated, but a warning is printed.

Value

A data frame with estimate the MASE and the number of runs used for the estimation for each index.

References

Carvalho, F., Winker, H., Courtney, D., Kapur, M., Kell, L., Cardinale, M., Schirripa, M., Kitakado, T., Yemane, D., Piner, K.R. Maunder, M.N., Taylor, I., Wetzel, C.R., Doering, K., Johnsonm, K.F., Methot, R. D. (2021). A cookbook for using model diagnostics in integrated stock assessments. Fisheries Research, 240, 105959.

Kell, L. T., Kimoto, A., & Kitakado, T. (2016). Evaluation of the prediction skill of stock assessment using hindcasting. Fisheries research, 183, 119-127.

Kell, L. T., Sharma, R., Kitakado, T., Winker, H., Mosqueira, I., Cardinale, M., & Fu, D. (2021). Validation of stock assessment methods: is it me or my model talking?. ICES Journal of Marine Science, 78(6), 2244-2255.

Winker, H., Carvalho, F., & Kapur, M. (2018). JABBA: just another Bayesian biomass assessment. Fisheries Research, 204, 275-288.

Examples

```
data(pol)
inp <- pol$albacore
rep <- fit.spict(inp)
rep <- hindcast(rep)
calc.mase(rep)
```

`calc.om` *Calculates the order of magnitude for the relative reference levels B/Bmsy and F/Fmsy*

Description

Calculates the order of magnitude for the relative reference levels B/Bmsy and F/Fmsy

Usage

```
calc.om(rep, CI = 0.95)
```

Arguments

`rep`	Result of fit.spict().
`CI`	Confidence intervals to be used for CI range, e.g. 0.9 for the 90% confidence intervals. By default (CI = 0.95), the 95% confidence intervals are used for the CI range.

Details

The lower, upper values and the CI range are based on the 95% confidence interval (CI; default).

Value

Matrix containing the order of magnitude for B/Bmsy and F/Fmsy.

calc.osa.resid *Calculate one-step-ahead residuals.*

Description

Calculate one-step-ahead residuals.

Usage

```
calc.osa.resid(rep)
```

Arguments

rep	A result report as generated by running fit.spict.

Details

In TMB one-step-ahead residuals are calculated by sequentially including one data point at a time while keeping the model parameters fixed at their ML estimates. The calculated residuals are tested for independence, bias, and normality.

Value

An updated result report, which contains one-step-ahead residuals stored in $osarC and $osarI.

Examples

```
data(pol)
rep <- fit.spict(pol$albacore)
rep <- calc.osa.resid(rep)
plotspict.osar(rep)
```

calc.process.resid *Calculate process residuals*

Description

Calculate process residuals

Usage

```
calc.process.resid(rep, dt = NULL)
```

Arguments

rep	A result report as generated by running `fit.spict`.
dt	Time resolution of process residuals. By default (NULL), the process residuals are calculated corresponding to the time resolution of the input data.

Details

Calculates process residuals for biomass and fishing mortality process.

Value

Data frame with year, process residuals for biomass and fishing mortality in the columns.

Examples

```
data(pol)
inp <- pol$albacore
rep <- fit.spict(inp)
rep <- calc.process.resid(rep)
plotspict.diagnostic.process(rep)
```

---

calc.tac *Calculate Total Allowable Catch (TAC)*

Description

Calculate Total Allowable Catch (TAC)

Usage

```
calc.tac(rep, inp = NULL, fractileCatch = 0.5, exp = TRUE)
```

Arguments

rep	A result report as generated by running fit.spict.
inp	Input list with ffac or catch observation corresponding to management. If NULL (default), the input list in rep is used.
fractileCatch	The fractile of the catch distribution to be used for setting the TAC. Default (0.5) corresponds to the median.
exp	Should tac be reported on natural scale? (default: TRUE)

Value

Total Allowable Catch (TAC)

check.catchList *Check catch list*

Description

Check catch list

Usage

```
check.catchList(catchList, sdfac = 1)
```

Arguments

catchList	List obtaining the elements 'obsC', 'timeC', and 'dtc' (optional element 'stdevfacC' which is 1 if not provided)
sdfac	Factor for the multiplication of the standard deviation of the catch (default: 1).

Details

Internal function that checks if catchList is complete.

Value

Checked catchList

check.ini *Check sensitivity of fit to initial parameter values*

Description

Check sensitivity of fit to initial parameter values

Usage

```
check.ini(input, ntrials = 10, verbose = TRUE, numdigits = 2)
```

Arguments

input	Either an inp list passing check.inp(), or a rep list where rep is the output of running fit.spict().
ntrials	The number of trials with different starting values to run.
verbose	If true write information to screen.
numdigits	Number of digits in reported results.

Value

List containing results of sensitivity check and associated initial values.

`check.inp` *Check list of input variables*

Description

Check list of input variables

Usage

```
check.inp(inp, verbose = TRUE, mancheck = TRUE)
```

Arguments

`inp`	List of input variables, see details for required variables.
`verbose`	Should detailed outputs be provided (default: TRUE).
`mancheck`	Should the time-dependent objects in `inp` be checked against the management time and corrected if necessary? (Default: TRUE)

Details

Fills in default values if missing.

Required inputs:

- "inp$obsC" Vector of catch observations.
- "inp$obsI and/or inp$obsE" List containing vectors of index observations and/or a vector of effort information.

Optional inputs:

- Data

- "inp$timeC" Vector of catch times. Default: even time steps starting at 1.
- "inp$timeI" List containing vectors of index times. Default: even time steps starting at 1.
- "inp$timeE" Vector of effort times. Default: even time steps starting at 1.
- "inp$dtc" Time interval for catches, e.g. for annual catches inp$dtc=1, for quarterly catches inp$dtc=0.25. Can be given as a scalar, which is then used for all catch observations. Can also be given as a vector specifying the catch interval of each catch observation. Default: min(diff(inp$timeC)).
- "inp$dte" Time interval for effort observations. For annual effort inp$dte=1, for quarterly effort inp$dte=0.25. Default: min(diff(inp$timeE)).
- "inp$nseasons" Number of within-year seasons in data. If inp$nseasons > 1 then a seasonal pattern is used in F. Valid values of inp$nseasons are 1, 2 or 4. Default: number of unique within-year time points present in data.
- "start.in.first.data.point" Logical. If `TRUE` (default), modelling time starts at the first available data point, otherwise it starts in the beginning of that year.

- Initial parameter values

  - "inpinilogn" Pella-Tomlinson exponent determining shape of production function. Default: log(2) corresponding to the Schaefer formulation.
  - "inpinilogm" Initial value for logm (log maximum sustainable yield). Default: log(mean(catch)).
  - "inpinilogK" Initial value for logK (log carrying capacity). Default: log(4*max(catch)).
  - "inpinilogq" Initial value for logq (log catchability of index). Default: log(max(index)/K).
  - "inpinilogsdb" Initial value for logsdb (log standard deviation of biomass process). Default: log(0.2).
  - "inpinilogsdf" Initial value for logsdf (log standard deviation of fishing mortality process). Default: log(0.2).
  - "inpinilogsdi" Initial value for logsdi (log standard deviation of index observation error). Default: log(0.2).
  - "inpinilogsdc" Initial value for logsdc (log standard deviation of catch observation error). Default: log(0.2).
  - "inpiniphi" Vector for cyclic B spline representing within-year seasonal variation. Default: rep(1, inp$nseasons).
  - "inpinilogsdu" Initial value for logsdu (log standard deviation of log U, the state of the coupled SDE representation of seasonality). Default: log(0.1).
  - "inpiniloglambda" Initial value for loglambda (log damping parameter of the coupled SDE representation of seasonality). Default: log(0.1).

- Initial values for unobserved states estimated as random effects

  - "inpinilogF" Log fishing mortality. Default: log(0.2*r), with r derived from m and K.
  - "inpinilogB" Log biomass. Default: log(0.5*K).
  - "inpinilogU" Log U, the state of the coupled SDE representation of seasonality. Default: log(1).

- Priors

Priors on model parameters are assumed generally assumed Gaussian and specified in a vector of length 2: c(log(mean), stdev in log domain, useflag [optional]). NOTE: if specifying a prior for a value in a temporal vector e.g. logB, then a fourth element is required specifying the year the prior should be applied. log(mean): log of the mean of the prior distribution. stdev in log: standard deviation of the prior distribution in log domain. useflag: if 1 then the prior is used, if 0 it is not used. Default is 1. To list parameters to which priors can be applied run list.possible.priors(). Example: intrinsic growth rate of 0.8 inp$priors$logr <- c(log(0.8), 0.1) inp$priors$logr <- c(log(0.8), 0.1, 1) # This includes the optional useflag Example: Biomass prior of 200 in 1985 inp$priors$logB <- c(log(200), 0.2, 1985) inp$priors$logB <- c(log(200), 0.2, 1, 1985) # This includes the optional useflag

- Settings/Options/Preferences

  - "inp$nspinup"Number of time steps used for spin-up (burn-in) period. Default: 0.
  - "inp$maninterval" Start and end time of management period. Default: One year interval starting at the beginning of the new year after the last observation. Example: inp$maninterval <- c(2020.25,2021.25)

- "inp$maneval" Time for the estimation of predicted model states (biomass and fishing mortality), which can be used to evaluate the implications of management scenarios. Default: At the end of the management interval `inp$maninterval[2]`. Example: inp$maneval <- 2021.25
- "inp$timepredc" Deprecated: Predict accummulated catch in the interval starting at $timepredc and $dtpredc into the future. Default depends on `inp$maninterval`.
- "inp$dtpredc" Deprecated: Length of catch prediction interval in years. Default depends on `inp$maninterval`.
- "inp$timepredi" Deprecated: Predict index until this time. Default depends on `inp$maneval`.
- "inp$manstart" Deprecated: Start of the management period. Updated argument `inp$maninterval`. Default depends on `inp$maninterval`.
- "inp$do.sd.report" Flag indicating whether SD report (uncertainty of derived quantities) should be calculated. For small values of inp$dteuler this may require a lot of memory. Default: TRUE.
- "inp$reportall" Flag indicating whether quantities derived from state vectors (e.g. B/Bmsy, F/Fmsy etc.) should be calculated by SD report. For small values of inp$dteuler (< 1/32) reporting all may have to be set to FALSE for sdreport to run. Additionally, if only reference points of parameter estimates are of interest one can set to FALSE to gain a speed-up. Default: TRUE.
- "inp$reportmode" Integer between 0 and 2 determining which objects will be adreported. Default: 0 = all quantities are adreported. Example: inp$reportmode <- 1
- "inp$reportRel" Flag indicating whether mean 1 standardized states (i.e. B/mean(B), F/mean(F) etc.) should be calculated by SD report. Default: FALSE.
- "inp$robflagc" Flag indicating whether robust estimation should be used for catches (either 0 or 1). Default: 0.
- "inp$robflagi" Vector of flags indicating whether robust estimation should be used for indices (either 0 or 1). Default: 0.
- "inp$ffac" Management scenario represented by a factor to multiply F with when calculating the F of the next time step. ffac=0.8 means a 20% reduction in F over the next year. The factor is only used when predicting beyond the data set. Default: 1 (0% reduction).
- "inp$dteuler" Length of Euler time step in years. Default: 1/16 year.
- "inp$phases" Phases can be used to fix/free parameters and estimate in different stages or phases. To fix e.g. logr at inpinilogr set inp$phases$logr <- -1. To free logalpha and estimate in phase 1 set inp$phases$logalpha <- 1.
- "inp$osar.method" Method to use in TMB's oneStepPredict function. Valid methods include: "oneStepGaussianOffMode", "fullGaussian", "oneStepGeneric", "oneStepGaussian", "cdf". See TMB help for more information. Default: "none" (i.e. don't run this).
- "inp$osar.trace" If TRUE print OSAR calculation progress to screen. Default: FALSE.
- "inp$osar.parallel" If TRUE parallelise OSAR calculation for speed-up. Default: FALSE.
- "inp$catchunit" Specify unit of catches to be used in plotting legends. Default: ".
- "inp$stdevfacC" Factors to multiply the observation error standard deviation of each individual catch observation. Can be used if some observations are more uncertain than others. Must be same length as observation vector. Default: 1.

- "inp$stdevfacI" Factors to multiply the observation error standard deviation of each individual index observation. Can be used if some observations are more uncertain than others. A list with vectors of same length as observation vectors. Default: 1.
- "inp$stdevfacE" Factors to multiply the observation error standard deviation of each individual effort observation. Can be used if some observations are more uncertain than others. A list with vectors of same length as observation vectors. Default: 1.
- "inp$mapsdi" Vector of length equal to the number of index series specifying which indices that should use the same sdi. For example: in case of 3 index series use `inp$mapsdi <- c(1, 1, 2)` to have series 1 and 2 share sdi and have a separate sdi for series 3. Default: 1:nindex, where nindex is number of index series.
- "inp$seasontype" If set to 1 use the spline-based representation of seasonality. If set to 2 use the oscillatory SDE system (this is more unstable and difficult to fit, but also more flexible).
- "inp$sim.random.effects"Should random effects (logB, logF, etc.) be simulated (default) or the same random effects be used (as specified in `inp$ini` or in an fitted spict object)?
- "inp$sim.fit"Should the estimated parameters from the last fit of a fitted spict object be used for simulation (`env$last.par`, default) or the inital values (specified in `inp$ini`)?. Note, that this only works if a fitted spict object is provided as input to `sim.spict`.

Value

An updated list of input variables checked for consistency and with defaults added.

Examples

```
data(pol)
(inp <- check.inp(pol$albacore))
```

`check.man` *Check the consistency of management scenarios in rep*

Description

Check the consistency of management scenarios in rep

Usage

```
check.man(
  rep,
  maninterval = NULL,
  maneval = NULL,
  verbose = TRUE,
  reportmode0 = TRUE
)
```

Arguments

rep	A result report as generated by running fit.spict.
maninterval	Two floats representing the start and end of the management period. Example: maninterval = c(2020.25,2021.25). Default: NULL.
maneval	Time at which to evaluate model states. Example: maneval = 2021.25. Default: NULL.
verbose	Should detailed outputs be provided (default: TRUE).
reportmode0	Should it be checked that the reportmode is 0 (default: TRUE).

Details

Internal function that checks if the fitted spict objects in rep$man have a consistent management interval.

Value

TRUE/FALSE

---

check.man.time *Checks and corrects management time to be within model time*

Description

Checks and corrects management time to be within model time

Usage

```
check.man.time(
  x,
  maninterval = NULL,
  maneval = NULL,
  verbose = TRUE,
  printTimeline = FALSE,
  mancheck = TRUE
)
```

Arguments

x	Either an input list from check.inp or a result report as generated by running fit.spict.
maninterval	Two floats representing the start and end of the management period. Example: maninterval = c(2020.25,2021.25). Default: NULL.
maneval	Time at which to evaluate model states. Example: maneval = 2021.25. Default: NULL.
verbose	Should detailed outputs be provided (default: TRUE).

mancheck	Should the time-dependent objects in inp be checked against the management time and corrected if necessary? (Default: TRUE)
printTimeLine	logical; print the management time line (default: FALSE)

Value

Updated input list or fitted spict object dependent on type of input.

Examples

```
data(pol)
inp <- check.inp(pol$albacore)
rep <- fit.spict(inp)

## with an input list
check.man.time(inp)

## with an output list
check.man.time(rep)
```

---

check.rep *Check rep list*

Description

Check rep list

Usage

```
check.rep(rep, reportmode0 = TRUE)
```

Arguments

rep	A result report as generated by running fit.spict.
reportmode0	Should it be checked that the reportmode is 0 (default: TRUE).

Details

Internal function that checks if rep is fitted spict object.

Value

Nothing

extract.hindcast.info *Extract hindcast info from a fitted spict object*

Description

Extract hindcast info from a fitted spict object

Usage

```
extract.hindcast.info(rep, CI = 0.95, verbose = TRUE)
```

Arguments

rep	Result of fit.spict that contains hindcasted runs added by hindcast.
CI	Confidence intervals to be calculated, e.g. 0.9 for the 90% confidence intervals. By default (CI = 0.95), the 95% confidence intervals are estimated.
verbose	Should detailed outputs be provided (default: TRUE).

Details

Note that a difference in the timing of the index observations of less than a month are considered acceptable for the estimation of the naive prediction residuals and no warning is printed. If the variability exceeds a month, the predictions are still calculated, but a warning is printed.

Value

A list with hindcast information.

extract.simstats *Extracts relevant statistics from the estimation of a simulated data set.*

Description

Extracts relevant statistics from the estimation of a simulated data set.

Usage

```
extract.simstats(rep, inp = NULL, exp = NULL, parnames = NULL)
```

Arguments

rep	A result report as generated by running fit.spict.
inp	The input list used as input to the validation.spict function.
exp	Should exp be taken of parameters?
parnames	Vector of parameter names to extract stats for.

Details

TBA

Value

A list containing the relevant statistics.

Examples

```
data(pol)
repin <- fit.spict(pol$albacore)
sim <- sim.spict(repin)
rep <- fit.spict(sim)
extract.simstats(rep)
```

`fd` *Format date*

Description

Format date

Usage

```
fd(d, dec = 2)
```

Arguments

`d`	Point in time in years as decimal number.
`dec`	Number of decimals.

Value

Correctly formatted date.

`fit.aspic` *Fits aspic to the data contained in the input file*

Description

Fits aspic to the data contained in the input file

Usage

```
fit.aspic(
  input,
  do.boot = FALSE,
  nboot = NULL,
  ciperc = NULL,
  verbose = FALSE,
  filebase = "tmp",
  savefile = NULL
)
```

Arguments

`input`	A spict input list containing observations.
`do.boot`	Do bootstrap to get uncertainties of estimates?
`nboot`	Number of bootstrap runs (only used if do.boot=TRUE). Prager suggests in the ASPIC manual p. 13 to use nboot > 1000 if ciperc > 80.
`ciperc`	Coverage percentage (integer between 0 and 100) of bootstrapped confidence intervals.
`verbose`	If TRUE write information to screen.
`filebase`	Basename of all generated aspic files.
`savefile`	Save results to this file.

Details

Only works on Linux. This furthermore requires that wine is installed and that aspic7 is installed and available to the PATH.

Value

List containing aspic results.

fit.jags *Fit the Meyer & Millar model using rjags*

Description

Fit the Meyer & Millar model using rjags

Usage

```
fit.jags(
  inp,
  fn,
  n.iter = 10000,
  n.chains = 1,
  burnin = round(n.iter/2),
  thin = 1000
)
```

Arguments

inp	Input list containing data and settings.
fn	Filename of containing BUGS code.
n.iter	Number of iterations.
n.chains	Number of chains.
burnin	Number of burn-in iterations.
thin	Thin chains by this value.

Value

The raw output of rjags::coda.samples.

fit.meyermillar *Fit the model of Meyer & Millar (1999)*

Description

Fit the model of Meyer & Millar (1999)

Usage

```
fit.meyermillar(mminp)
```

Arguments

mminp	Input list similar to the input to fit.spict()

Details

Same input structure as for fit.spict(). Fitting the model of Meyer & Millar requires the packages rjags and coda. It furthermore requires that priors are specified for K, r, q, sigma2 (process error variance) and tau2 (observation error variance). Following Meyer & Millar (1999) the priors are:

- "K" log-normal.
- "r" log-normal.
- "q" inverse-gamma.
- "tau2" inverse-gamma.
- "sigma2" inverse-gamma.

See example for how to specify priors.

Value

List containing results

Examples

```
## Not run:
priors <- list()
priors$K <- c(5.042905, 3.76)
priors$r <- c(-1.38, 3.845)
priors$iq <- c(0.001, 0.0012)
priors$itau2 <- c(1.709, 0.00861342)
priors$isigma2 <- c(3.785518, 0.0102232)
priors$logPini <- -0.223
data(pol)
inp <- pol$albacore
inp$meyermillar$n.iter <- 10000
inp$meyermillar$burnin <- 1000
inp$meyermillar$thin <- 10
inp$meyermillar$n.chains <- 1
inp$meyermillar$priors <- priors
res <- fit.meyermillar(inp)
summary(res$jags)

## End(Not run)
```

---

fit.spict *Fit a continuous-time surplus production model to data.*

Description

Fit a continuous-time surplus production model to data.

Usage

```
fit.spict(inp, verbose = TRUE, dbg = 0)
```

Arguments

inp	List of input variables as output by check.inp.
verbose	Should detailed outputs be provided (default: TRUE).
dbg	Debugging option. Will print out runtime information useful for debugging if set to 1. Will print even more if set to 2.

Details

Fits the model using the TMB package and returns a result report containing estimates of model parameters, random effects (biomass and fishing mortality), reference points (Fmsy, Bmsy, MSY) including uncertainties given as standard deviations.

Model parameters using the formulation of Fletcher (1978):

- "logn" Parameter determining the shape of the production curve as in the generalised form of Pella & Tomlinson (1969).
- "logm" Log of maximum sustainable yield.
- "logK" Log of carrying capacity.
- "logq" Log of catchability vector.
- "logsdb" Log of standard deviation of biomass process error.
- "logsdf" Log of standard deviation of fishing mortality process error.
- "logsdi" Log of standard deviation of index observation error.
- "logsdc" Log of standard deviation of catch observation error.

Unobserved states estimated as random effects:

- "logB" Log of the biomass process given by the stochastic differential equation: dB_t = r*B_t*(1-(B_t/K)^n)*dt + sdb*dW_t, where dW_t is Brownian motion.
- "logF" Log of the fishing mortality process given by: dlog(F_t) = f(t, sdf), where the function f depends on the choice of seasonal model.

Other parameters (which are only needed in certain cases):

- "logphi" Log of parameters used to specify the cyclic B spline representing seasonal variation. Used when inp$nseasons > 1 and inp$seasontype = 1.
- "logU" Log of the state of the coupled SDE system used to represent seasonal variation, i.e. when inp$nseasons > 1 and inp$seasontype = 2.
- "loglambda" Log of damping parameter when using the coupled SDE system to represent seasonal variation, i.e. when inp$nseasons > 1 and inp$seasontype = 2.
- "logsdu" Log of standard deviation of process error of U_t (the state of the coupled SDE system) used to represent seasonal variation, i.e. when inp$nseasons > 1 and inp$seasontype = 2.
- "logsde" Log of standard deviation of observation error of effort data. Only used if effort data is part of input.
- "logp1robfac" Log plus one of the coefficient to the standard deviation of the observation error when using a mixture distribution robust toward outliers, i.e. when either inp$robflag = 1 and/or inp$robflagi = 1.

- "logitpp"Logit of the proportion of narrow distribution when using a mixture distribution robust toward outliers, i.e. when either inp$robflag = 1 and/or inp$robflagi = 1.

Parameters that can be derived from model parameters:

- "logr" Log of intrinsic growth rate (r = 4m/K).
- "logalpha" Proportionality factor for the observation noise of the indices and the biomass process noise: sdi = exp(logalpha)*sdb. (normally set to logalpha=0)
- "logbeta" Proportionality factor for the observation noise of the catches and the fishing mortality process noise: sdc = exp(logbeta)*sdf. (this is often difficult to estimate and can result in divergence of the optimisation. Normally set to logbeta=0)
- "logBmsy" Log of the equilibrium biomass (Bmsy) when fished at Fmsy.
- "logFmsy" Log of the fishing mortality (Fmsy) leading to the maximum sustainable yield.
- "MSY" The yield when the biomass is at Bmsy and the fishing mortality is at Fmsy, i.e. the maximum sustainable yield.

The above parameter values can be extracted from the fit.spict() results using get.par().

Model assumptions

1. The intrinsic growth rate (r) represents a combination of natural mortality, growth, and recruitment.
2. The biomass B_t refers to the exploitable part of the stock. Estimates in absolute numbers (K, Bmsy, etc.) should be interpreted in light of this.
3. The stock is closed to migration.
4. Age and size-distribution are stable in time.
5. Constant catchability of the gear used to gather information for the biomass index.

Value

A result report containing estimates of model parameters, random effects (biomass and fishing mortality), reference points (Fmsy, Bmsy, MSY) including uncertainties given as standard deviations.

Examples

```
data(pol)
rep <- fit.spict(pol$albacore)
Bmsy <- get.par('logBmsy', rep, exp=TRUE)
summary(rep)
plot(rep)
```

get.AIC *Calculate AIC from a rep list.*

Description

Calculate AIC from a rep list.

Usage

```
get.AIC(rep)
```

Arguments

rep	A result report as generated by running fit.spict.

Value

AIC

get.catchindexoverlap *Find observations of catch and index that overlap*

Description

Find observations of catch and index that overlap

Usage

```
get.catchindexoverlap(inp)
```

Arguments

inp	An input list containing data.

Value

List containing overlapping catch (y) and index (z) observations and their time vectors.

`get.colnms` *Get column names for data.frames.*

Description

Get column names for data.frames.

Usage

```
get.colnms()
```

Value

Vector containing column names of data frames.

`get.cov` *Get covariance matrix of two reported quantities not of fixed model parameters. Covariance of fixed model parameters can be found in rep$cov.fixed.*

Description

Get covariance matrix of two reported quantities not of fixed model parameters. Covariance of fixed model parameters can be found in rep$cov.fixed.

Usage

```
get.cov(rep, parname1, parname2, cor = FALSE)
```

Arguments

`rep`	Result of fit.spict().
`parname1`	Name first parameter.
`parname2`	Name second parameter.
`cor`	If TRUE correlation matrix is reported instead of covariance matrix

Value

Covariance matrix of specified parameters.

get.EBinf — *Calculate E(Binfinity) the fished equilibrium.*

Description

Calculate E(Binfinity) the fished equilibrium.

Usage

```
get.EBinf(rep)
```

Arguments

rep	A result of fit.spict.

Details

If a seasonal pattern in F is imposed the annual average F is used for calculating the expectation.

Value

E(Binf).

get.ffac — *Estimate fishing mortality factor minimising probability of specified model variable hitting a specified reference level under given fishing mortality*

Description

Estimate fishing mortality factor minimising probability of specified model variable hitting a specified reference level under given fishing mortality

Usage

```
get.ffac(
  rep,
  var = "logBpBmsy",
  ref = 1,
  problevel = 0.95,
  reportmode = 1,
  verbose = TRUE
)
```

Arguments

rep	A result report as generated by running fit.spict.
var	A variable of the spict model (default: "logBpBmsy").
ref	Reference level relative to specified variable (default: 1)
problevel	Probability level of the risk aversion (default: 0.95).
reportmode	Integer between 0 and 2 determining which objects will be adreported (default: 1).
verbose	logical; print informative text (default: TRUE).

Value

Optimised Fishing mortality for P(Bp<Blim)

---

get.manC *Estimate catch for management period based on last catch observations*

Description

Estimate catch for management period based on last catch observations

Usage

```
get.manC(rep, inp)
```

Arguments

rep	A result report as generated by running fit.spict.
inp	Input list with ffac or catch observation corresponding to management.

Details

Internal function that estimates the catch in the management period based on the catch observations in the last year. Only catch observations in the last year are considered. If the management period is longer than a year the catches of the last year are raised. If the management period is shorter than a year, but only annual catches are available, the respective fraction of the last annual catch observation is used. If both the management period and the catch observations are subannual, the subannual catches of the respective 'seasons' of the year corresponding to the 'season' of the management period are used. Be aware that the estimated catch might correspond to a different season(s) than the management period both are subannual and some catch observations are missing.

Value

Table with catches for management period based on last observed catches and corresponding times.

get.manlimits *Get limts of any parameter considering all spict objects in rep$man*

Description

Get limts of any parameter considering all spict objects in rep$man

Usage

```
get.manlimits(rep, par, CI = 0.95, with.spinup = FALSE)
```

Arguments

rep	A result report as generated by running fit.spict.
par	The name of the parameter to be plotted.
CI	Confidence intervals to be calculated, e.g. 0.9 for the 90% confidence intervals. By default (CI = 0.95), the 95% confidence intervals are estimated.
with.spinup	If TRUE returns the parameter limits considering the spinup period? Default: FALSE.

Details

‘with.spinup‘ argument only works with ‘par = "time"‘.

Value

plotting limits for all reps in rep$man

get.manmax *Get spict object in rep$man with longest time series*

Description

Get spict object in rep$man with longest time series

Usage

```
get.manmax(rep)
```

Arguments

rep	A result report as generated by running fit.spict.

Value

rep in rep$man which hast the longest time series

get.mfrow — *Get mfrow from the number of plots to be plotted*

Description

Get mfrow from the number of plots to be plotted

Usage

```
get.mfrow(n)
```

Arguments

n	Number of plots to be plotted.

Value

Nothing

get.msyvec — *If multiple growth rates (r) are used (e.g. for a seasonal model), return specified reference point for all instances of r.*

Description

If multiple growth rates (r) are used (e.g. for a seasonal model), return specified reference point for all instances of r.

Usage

```
get.msyvec(inp, msy)
```

Arguments

inp	An input list as validated by check.inp().
msy	Matrix containing reference point values as given by get.par().

Value

A list containing reference point estimates with upper and lower CI bounds.

get.no.active.priors *Get number of active priors*

Description

Get number of active priors

Usage

```
get.no.active.priors(inp)
```

Arguments

inp	An input list containing priors (after call to check.inp and/or fit.spict)

Value

number of active priors

get.order *Get order of printed quantities.*

Description

Get order of printed quantities.

Usage

```
get.order()
```

Value

Vector containing indices of printed quantities.

get.osar.pvals *Check whether ACF of catch and index residuals is significant in any lags.*

Description

Check whether ACF of catch and index residuals is significant in any lags.

Usage

```
get.osar.pvals(rep)
```

Arguments

rep	Result of fit.spict(), but requires that also residuals have been calculated using calc.osa.resic().

Value

Vector of p-values of length equal to the number of data series.

get.par *Extract parameters from a result report as generated by fit.spict.*

Description

Extract parameters from a result report as generated by fit.spict.

Usage

```
get.par(
  parname,
  rep = rep,
  exp = FALSE,
  random = FALSE,
  fixed = FALSE,
  CI = 0.95
)

list.quantities(rep)
```

Arguments

parname	Character string containing the name of the variable of interest.
rep	A result report as generated by running fit.spict.
exp	Take exp of the variable? TRUE/FALSE.
random	DUMMY not used anymore. (Is the variable a random effect? TRUE/FALSE.)
fixed	DUMMY not used anymore. (Is the variable a fixed effect? TRUE/FALSE.)
CI	Confidence intervals to be calculated, e.g. 0.9 for the 90% confidence intervals. By default (CI = 0.95), the 95% confidence intervals are estimated.

Details

get.par is a helper function for extracting the value and uncertainty of a specific model parameter, random effect or derived quantity. list.quantities gives the names of all quantities.

Value

get.par returns a matrix with four columns containing respectively: 1) the lower 95% confidence limit; 2) the parameter estimate; 3) the upper 95% confidence limit; 4) the parameter standard deviation in the domain it was estimated (log or non-log). ‘list.quantities‘ returns a vector with the names of all estimated parameters and derived quantities.

Examples

```
## Run the South Atlantic albacore assessment
data(pol)
rep <- fit.spict(pol$albacore)

## See all quantitites that can be extracted
list.quantities(rep)

## Extract the Bmsy reference point
Bmsy <- get.par('logBmsy', rep, exp=TRUE)

## Extract the exploitable biomass estimates
Best <- get.par('logB', rep, exp=TRUE)

## Extract the estimated carrying capacity
K <- get.par('logK', rep, exp=TRUE)
```

---

get.spline *Get the values of the seasonal spline for F.*

Description

Get the values of the seasonal spline for F.

Usage

```
get.spline(logphi, order, dtfine = 1/100)
```

Arguments

logphi	Values of the phi vector.
order	Order of the spline.
dtfine	Time between points where spline is evaluated.

Value

Spline values at the points between 0 and 1 with dtfine as time step.

---

get.version *Get version of spict including git sha1 version if available.*

Description

Get version of spict including git sha1 version if available.

Usage

```
get.version(pkg = "spict")
```

Arguments

pkg	Name of package.

Value

Package version

---

guess.m *Use a simple linear regression to guess m (MSY).*

Description

Use a simple linear regression to guess m (MSY).

Usage

```
guess.m(inp, all.return = FALSE)
```

Arguments

inp	An input list containing data.
all.return	If true also return a guess on Emsy (effort at MSY) and components of the linear regression.

Details

Equations 9.1.7 and 9.1.8 on page 284 of FAO's tropical assessment book are used to guess MSY.

Value

The guess on MSY.

---

hindcast *Conduct hindcasting analysis*

Description

Conduct hindcasting analysis

Usage

```
hindcast(
  rep,
  npeels = 7,
  reduce.output.size = TRUE,
  mc.cores = 1,
  peel.dtc = FALSE
)
```

Arguments

rep	rep Result of fit.spict.
npeels	Number of years/seasons (dependent on dtc) of data (catch and effort) to remove (this is also the total number of model runs).
reduce.output.size	logical, if TRUE (default) hindcasting is run with getReportCovariance and getJointPrecision set as FALSE
mc.cores	Number of cores for parallel::mclapply function. By default 1.
peel.dtc	Peel according to catch seasons (dtc) rather than years? It only differs if the data includes seasonal catches (Default: FALSE)

Details

This method creates a number of subsets (or peels) specified with the argument `npeels` by sequentially omitting all observations from the most recent time step (by default corresponding to a year). Then, spict is fitted to each subset while excluding all index observations in the most recent year with data. The resulting fitted spict objects are attached to the base spict object as a list element labeled `hindcast` and resulted by this method.

The prediction of the excluded index observations relative to the actual excluded index can then be compared to a 'naive' prediction using the preceding index observation. This allows to estimate the Mean Absolute Scaled Error (MASE) for each index `calc.mase`.

The predicted indices can be visualised with the function `plotspict.hindcast`.

Value

A `spictcls` list with the added element `hindcast` containing the results of the hindcasting analysis. Use `plotspict.hindcast` to plot these results.

References

Carvalho, F., Winker, H., Courtney, D., Kapur, M., Kell, L., Cardinale, M., Schirripa, M., Kitakado, T., Yemane, D., Piner, K.R. Maunder, M.N., Taylor, I., Wetzel, C.R., Doering, K., Johnsonm, K.F., Methot, R. D. (2021). A cookbook for using model diagnostics in integrated stock assessments. Fisheries Research, 240, 105959.

Kell, L. T., Kimoto, A., & Kitakado, T. (2016). Evaluation of the prediction skill of stock assessment using hindcasting. Fisheries research, 183, 119-127.

Kell, L. T., Sharma, R., Kitakado, T., Winker, H., Mosqueira, I., Cardinale, M., & Fu, D. (2021). Validation of stock assessment methods: is it me or my model talking?. ICES Journal of Marine Science, 78(6), 2244-2255.

Winker, H., Carvalho, F., & Kapur, M. (2018). JABBA: just another Bayesian biomass assessment. Fisheries Research, 204, 275-288.

Examples

```
data(pol)
inp <- pol$albacore
rep <- fit.spict(inp)
rep <- hindcast(rep, npeels = 5)
plotspict.hindcast(rep)
```

`invlogit` *Inverse logit transform.*

Description

Inverse logit transform.

Usage

```
invlogit(a)
```

Arguments

a	Value to take inverse logit of.

Value

Inverse logit.

---

invlogp1 *Inverse log "plus one" transform*

Description

Inverse log "plus one" transform

Usage

```
invlogp1(a)
```

Arguments

a	Value to take inverse logp1 of.

Details

If a = log(b-1), then the inverse transform is b = 1 + exp(a). Useful for values with lower bound at 1.

Value

Inverse logp1.

latex.figure *Generate latex code for including a figure.*

Description

Generate latex code for including a figure.

Usage

```
latex.figure(figfile, reportfile, caption = "")
```

Arguments

figfile	Path to figure file.
reportfile	Path to report file.
caption	This character string will be included as the figure caption.

Value

Nothing.

likprof.spict *Create profile likelihood*

Description

Create profile likelihood

Usage

```
likprof.spict(input, verbose = FALSE, mc.cores = 1)
```

Arguments

input	A list containing observations and initial values for non profiled parameters (essentially an inp list) with the additional key "likprof" (see details for required keys). A valid result from fit.spict() containing an "inp" key with the described properties is also accepted.
verbose	Print progress to screen.
mc.cores	Number of cores for parallel::mclapply function. By default 1.

Details

The "likprof" list must containg the following keys:

- "pars" A character vector of length equal 1 or 2 containing the name(s) of the parameters to calculate the profile likelihood for.
- "parrange" A vector containing the parameter range(s) to profile over: parrange = c(min(par1), max(par1), min(par2), max(par2)).

Optional:

- "nogridpoints" Number of grid points to evaluate the profile likelihood for each parameter. Default: 9. Note: with two parameters the calculation time increases quadratically when increasing the number of gridpoints.

Value

The output is the input with the likelihood profile information added to the likprof key of either inp or rep$inp.

Examples

```
data(pol)
inp <- pol$albacore
inp$likprof <- list()
inp$likprof$pars <- 'logK'
inp$likprof$parrange <- c(log(80), log(400))
inp$likprof$nogridpoints <- 15
rep <- fit.spict(inp)
rep <- likprof.spict(rep)
plotspict.likprof(rep, logpar=TRUE)
```

---

list.possible.priors *List parameters to which priors can be added*

Description

List parameters to which priors can be added

Usage

```
list.possible.priors()
```

Value

Prints parameters to which priors can be added.

make.datin *Create data list used as input to TMB::MakeADFun.*

Description

Create data list used as input to TMB::MakeADFun.

Usage

```
make.datin(inp, dbg = 0)
```

Arguments

inp	List of input variables as output by check.inp.
dbg	Debugging option. Will print out runtime information useful for debugging if set to 1.

Value

List to be used as data input to TMB::MakeADFun.

make.fconvec *Make fcon vector*

Description

Make fcon vector

Usage

```
make.fconvec(inp, fcon)
```

Arguments

inp	Input list
fcon	Constant to add to F

Value

Input list containing fconvec

make.ffacvec *Make ffac vector*

Description

Make ffac vector

Usage

```
make.ffacvec(inp, ffac)
```

Arguments

inp	Input list
ffac	Factor to multiply current F by

Value

Input list containing ffacvec

make.obj *Create TMB obj using TMB::MakeADFun and squelch screen printing.*

Description

Create TMB obj using TMB::MakeADFun and squelch screen printing.

Usage

```
make.obj(datin, pl, inp, phase = 1)
```

Arguments

datin	Data list.
pl	Parameter list.
inp	List of input variables as output by check.inp.
phase	Estimation phase, integer.

Value

List to be used as data input to TMB.

`make.report` *Creates a pdf file containing the summary output and result plots*

Description

Creates a pdf file containing the summary output and result plots

Usage

```
make.report(
  rep,
  reporttitle = "",
  reportfile = "report.tex",
  summaryoutfile = "summaryout.txt",
  keep.figurefiles = FALSE,
  keep.txtfiles = FALSE,
  keep.texfiles = FALSE
)
```

Arguments

`rep`	A valid result from fit.spict with OSA residuals.
`reporttitle`	This character string will be printed as the first line of the report.
`reportfile`	A connection, or a character string naming the file ('.tex' file) to print to. If not a connection, `make.report` prints to the working directory (default).
`summaryoutfile`	Summary output filename.
`keep.figurefiles`	If TRUE generated figure files will not be cleaned up.
`keep.txtfiles`	If TRUE generated txt files will not be cleaned up.
`keep.texfiles`	If TRUE generated tex file will not be cleaned up.

Details

This function probably requires that you are running linux and that you have latex functions installed (pdflatex).

Value

Nothing.

make.rpellipse *Calculate confidence ellipsis for reference points.*

Description

Calculate confidence ellipsis for reference points.

Usage

```
make.rpellipse(rep)
```

Arguments

rep	A result report as generated by running fit.spict.

Details

Calculates the confidence ellipsis of logBmsy and logFmsy (last if multiple)

Value

A matrix with two columns containing the x and y coordinates of the ellipsis.

make.splinemat *Make a spline design matrix*

Description

Make a spline design matrix

Usage

```
make.splinemat(nseasons, order, dtfine = 1/100)
```

Arguments

nseasons	Number of seasons
order	Order of the spline
dtfine	Time between points where spline is evaluated

Value

Spline design matrix.

man.cols *Load color of management scenarios.*

Description

Load color of management scenarios.

Usage

```
man.cols()
```

Value

Color vector

man.select *Select management scenarios*

Description

Select management scenarios

Usage

```
man.select(rep, scenarios = "all", spictcls = FALSE, verbose = TRUE)
```

Arguments

rep	A result report as generated by running manage or add.man.scenario.
scenarios	Selection of scenarios in preferred order. Can be a vector with the names of the selected scenarios or numbers indicating their position in rep$man, e.g. c(6,2) for the 6th and 2nd scenario in the rep$man list. Setting this argument to NULL or "none", removes all scenarios from the spict object. By default ('all'), all scenarios are selected.
spictcls	Should selected scenario be a standard spictcls object? Default is FALSE. See details for more information.
verbose	Should detailed outputs be provided (default: TRUE).

Value

A fitted spict object wit selected management scenarios in preferred order in rep$man. This function can also be used to select a specific scenarios in rep$man as the new main spictcls object. By setting the argument spictcls to TRUE, management related catch observations are removed and the retaped spict object of class 'spictcls' is returned, comparable to the object returned by fit.spict. This only works if one scenario is selected (length(scnearios) == 1).

Examples

```
data(pol)
rep <- fit.spict(pol$albacore)
rep <- manage(rep, c(2,4,6))

## based on names
names(rep$man)
rep1 <- man.select(rep, c("currentF","noF"))

## based on indices
length(rep$man)
rep2 <- man.select(rep, c(1,3))

## select specific scenario as new spictcls object
rep3 <- man.select(rep, 1, spictcls = TRUE)
```

`man.tac` *Get the TAC for the management scenarios*

Description

Get the TAC for the management scenarios

Usage

```
man.tac(rep, fractileCatch = 0.5, exp = TRUE, verbose = TRUE)
```

Arguments

`rep`	A result report as generated by running `manage` or `add.man.scenario`.
`fractileCatch`	Fractile of predicted catch distribution. By default (0.5), the median is being used.
`exp`	Should tac be reported on natural scale (default: TRUE).
`verbose`	Should detailed outputs be provided (default: TRUE).

Value

rep wit selected management scenarios

Examples

```
data(pol)
rep <- fit.spict(pol$albacore)
rep <- manage(rep, c(3,4,5))

## Median of predicted catch distributions
man.tac(rep)
```

```
## 30th percentile of catch distributions
man.tac(rep, fractileCatch = 0.3)
```

`man.timeline` *Print a schematic to the console visualising the management timeline*

Description

Print a schematic to the console visualising the management timeline

Usage

```
man.timeline(x, verbose = TRUE, obsonly = FALSE)
```

Arguments

`x`	Either an input list from `check.inp` or a result report as generated by running `fit.spict`.
`verbose`	Should detailed outputs be provided (default: TRUE).
`obsonly`	Display observation period only

Value

Nothing

Examples

```
data(pol)
inp <- check.inp(pol$albacore)
inp$maninterval <- c(1991,1992)
rep <- fit.spict(inp)

## based on an input list
man.timeline(inp)

## based on an output list
man.timeline(rep)
```

manage *Calculate predictions under 8 default management scenarios*

Description

Calculate predictions under 8 default management scenarios

Usage

```
manage(
  rep,
  scenarios = "all",
  maninterval = NULL,
  maneval = NULL,
  intermediatePeriodCatch = NULL,
  intermediatePeriodCatchSDFac = 1,
  intermediatePeriodCatchList = NULL,
  verbose = TRUE,
  dbg = 0
)
```

Arguments

rep	A result report as generated by running `fit.spict`.
scenarios	Vector of integers specifying which scenarios to run or 'all' to run all scenarios. Default: 'all'.
maninterval	Two floats representing the start and end of the management period. Example: maninterval = c(2020.25,2021.25). Default: NULL.
maneval	Time at which to evaluate model states. Example: maneval = 2021.25. Default: NULL.
intermediatePeriodCatch	Catch during intermediate period, e.g. last year's TAC (default: `NULL`; see details for more information).
intermediatePeriodCatchSDFac	Factor for the multiplication of the standard deviation of the catch during the intermediate period (default: 1).
intermediatePeriodCatchList	List defining catch in the intermediate period obtaining the elements 'obsC', 'timeC', and 'dtc' (optional element 'stdevfacC' which is 1 if not provided)
verbose	Should detailed outputs be provided (default: TRUE).
dbg	Debug flag, dbg=1 some output, dbg=2 more output.

Details

The 8 default scenarios are:

"1" "currentCatch": Keep the catch of the current year (i.e. the last observed catch).

"2" "currentF": Keep the F of the current year.

"3" "Fmsy": Fish at Fmsy i.e. F=Fmsy.

"4" "noF": No fishing, reduce to 1% of current F.

"5" "reduceF25": Reduce F by X%. Default X = 25.

"6" "increaseF25": Increase F by X%. Default X = 25.

"7" "msyHockeyStick": Use ICES MSY hockey-stick advice rule (ICES, 2017).

"8" "ices": Use ICES MSY hockey-stick advice rule with Blim and the 0.35 catch fractile (ICES, 2025).

Scenario 7 implements the ICES MSY advice rule for stocks that are assessed using spict (ICES 2017). MSY B_trigger is set equal to B_MSY / 2. Then fishing mortality in the short forecast is calculated as:

F_y+1 = F_y * min 1, median[B_y+1 / MSY B_trigger] / median[F_y/F_MSY]

Scenario 8 is similar to scenario 7, but includes an additional biomass limit reference points B_lim = 0.3 B_MSY and accounts for assessment uncertainty by using the 35th percentile of the distribution of the predicted catch. If B(y) is below B_lim, F(y+1) is set to 0.

Dependent on the start of the management period (e.g. advice year), there might be a time lag between the last observation and the start of the management period, often referred to as the intermediate period. If this is the case, an assumption about the catch during intermediate time period (e.g. assessment year) has to be made. Two meaningful assumptions are:

1 The catch in the intermediate period is based on the fishing mortality which is extrapolated from the previous year. This is the default assumption;

2 The catch in the intermediate period is directly specified. This could for example be the TAC recommended in the previous year. The catch can be specified by means of the argument `intermediatePeriodCatch`. Be aware that this catch might correspond to several years or a fraction of a year depending on the time between the last observation and the start of the management period. The function `man.timeline` can help visualising the default or specified intermediate period in your data. The argument `intermediatePeriodCatchSDFac` allows to specify the factor with which to multiply the standard deviation of the catch (σ_C) with. It is thus a measure of the certainty around the catch in the intermediate period. The argument `intermediatePeriodCatchList` allows to define a list with catches and their intervals. It is a list with the elements 'obsC', 'timeC', 'dtc' and the optional element 'stdevfacC' (which is equal to 1 if not provided).

Value

List containing results of management calculations.

References

ICES. 2017. Report of the Workshop on the Development of the ICES approach to providing MSY advice for category 3 and 4 stocks (WKMSYCat34), 6-10 March 2017, Copenhagen, Denmark. ICES CM 2017/ACOM:47. 53 pp.

Examples

```
data(pol)
rep <- fit.spict(pol$albacore)
repman <- manage(rep, c(2,4,8))
sumspict.manage(repman) # To print projections
```

---

meanvar2shaperate *Convert mean and variance to shape and rate of gamma distribution*

Description

Convert mean and variance to shape and rate of gamma distribution

Usage

```
meanvar2shaperate(mean, var)
```

Arguments

mean	Mean value.
var	Variance.

Value

Vector containing shape and rate parameters.

---

modefrac2shaperate *Convert mode and fractile to shape and rate in Gamma distribution (only for mode>0, i.e. shape >= 1)*

Description

Convert mode and fractile to shape and rate in Gamma distribution (only for mode>0, i.e. shape >= 1)

Usage

```
modefrac2shaperate(mode, xf, f = 0.9)
```

Arguments

mode	x for which f(x) is at the maximum
xf	x for which P(X<=x) = f
f	fractile

Value

Vector containing shape and rate parameters.

---

mohns_rho *Calculate Mohn's rho for different estimates*

Description

Calculate Mohn's rho for different estimates

Usage

```
mohns_rho(rep, what = c("FFmsy", "BBmsy"), annualfunc = mean)
```

Arguments

rep	A valid result from fit.spict
what	character vector specifying the quantities
annualfunc	function used to convert subannual data into annual

Details

A function that calculates Mohn's rho for selected estimated quantities. The function allows the user to define the method of aggrgating from the subannual time steps (1/dteuler) into annual values; the default is to take the mean.

Value

A named vector with the Monh's rho value for each quantity.

Examples

```
data(pol)
inp <- pol$albacore
rep <- fit.spict(inp)
rep <- retro(rep, nretroyear = 4)
mohns_rho(rep)
```

osar.acf.plot *Plot osar acf*

Description

Plot osar acf

Usage

```
osar.acf.plot(res, lag.max, pval, ylab)
```

Arguments

res	Residuals
lag.max	Maximum lag to use in acf calculations.
pval	P value
ylab	Y-axis label

Value

Nothing.

osar.qq.plot *Plot osar qq*

Description

Plot osar qq

Usage

```
osar.qq.plot(res, pval)
```

Arguments

res	Residuals
pval	P value

Value

Nothing.

plot.col *Plot model points colored depending on the quarter to which they belong.*

Description

Plot model points colored depending on the quarter to which they belong.

Usage

```
## S3 method for class 'col'
plot(
  time,
  obs,
  obsx = NULL,
  pch = 1,
  add = FALSE,
  typ = "p",
  do.line = TRUE,
  add.legend = FALSE,
  add.vline.at = NULL,
  ...
)
```

Arguments

time	Time vector.
obs	Observation vector (or residual vector).
obsx	Second observation vector for use as independent variable instead of time.
pch	Point character.
add	If TRUE plot is added to the current plot.
typ	Plot type.
do.line	If TRUE draw a line between points.
add.legend	If TRUE add legend containing information on quarters.
add.vline.at	If not NULL will draw a vertical line at the given time point.
...	Additional plotting arguments.

Value

Nothing.

`plot.spictcls` *Plot summarising spict results.*

Description

Plot summarising spict results.

Usage

```
## S3 method for class 'spictcls'
plot(
  x,
  stamp = get.version(),
  verbose = TRUE,
  CI = 0.95,
  plot.spinup = FALSE,
  ...
)
```

Arguments

`x`	A result report as generated by running fit.spict.
`stamp`	Stamp plot with this character string.
`verbose`	Should detailed outputs be provided (default: TRUE).
`CI`	Confidence intervals to be calculated, e.g. 0.9 for the 90% confidence intervals. By default (CI = 0.95), the 95% confidence intervals are estimated.
`plot.spinup`	If TRUE spinup period is plotted, if spinup was used (inp$nspinup > 0). Default: FALSE.
`...`	additional arguments affecting the summary produced.

Details

Create a plot containing the following:

- 1. Estimated biomass using plotspict.biomass().
- 2. Estimated fishing mortality using plotspict.f().
- 3. Observed versus predicted catches using plotspict.catch().
- 4. Estimated biomass relative to Bmsy using plotspict.bbmsy().
- 5. Estimated fishing mortality relative to Fmsy using plotspict.ffmsy().
- 6. Estimated F versus estimated B using plotspict.fb().
- 7. Observed versus theoretical production using plotspict.production().

Optional plots included if relevant:

- Estimated seasonal spline using plotspict.season().

- Calculated time-constant using plotspict.tc().
- First prior and corresponding posterior distribution using plotspict.priors().
- One-step-ahead residuals of catches using plotspict.osar().
- One-step-ahead residuals of catches using plotspict.osar().

If no management scenarios are included in `rep$man`, the grey vertical line corresponds to the time of the last observation. If management scenarios are included in `rep$man`, the prediction and confidence intervals of the base scenario (`rep`) are omitted and instead the projections of the different management scenarios are drawn in different colours. Dotted lines of the management scenarios reflect the intermediate period, while solid lines reflect the management period. Additionally, two vertical lines correspond to the start and end of the management period.

Be aware that potential catch intervals of more a year, e.g. biennial assessment so that the intermediate period spans two years, or management period spans two years, are equally split up into annual intervals.

Be aware of the fact that the catches represent intervals, where the length of the interval is indicated by `dtc`, e.g. with $dtc = 1, C(1990) = [1990, 1990[$. In the plot the catches (and vertical lines) correspond to the beginning of the catch interval. It might thus seem as if the time of the vertical lines and the management interval would not align.

Value

Nothing.

Examples

```
data(pol)
rep <- fit.spict(pol$albacore)
plot(rep)
```

`plot2` *Plot summarising spict results (alternative plot composition)*

Description

Plot summarising spict results (alternative plot composition)

Usage

```
plot2(rep, stamp = get.version(), verbose = TRUE, CI = 0.95, ...)
```

Arguments

rep	A result report as generated by running fit.spict.
stamp	Stamp plot with this character string.
verbose	Should detailed outputs be provided (default: TRUE).
CI	Confidence intervals to be calculated, e.g. 0.9 for the 90% confidence intervals. By default (CI = 0.95), the 95% confidence intervals are estimated.
...	additional arguments affecting the summary produced.

Details

Create a plot containing the following:

- 1. Estimated biomass relative to Bmsy using plotspict.bbmsy().
- 2. Estimated fishing mortality relative to Fmsy using plotspict.ffmsy().
- 3. Observed versus predicted catches using plotspict.catch().
- 4. Estimated F versus estimated B using plotspict.fb().

If no management scenarios are included in rep$man, the grey vertical line corresponds to the time of the last observation. If management scenarios are included in rep$man, the prediction and confidence intervals of the base scenario (rep) are omitted and instead the projections of the different management scenarios are drawn in different colours. Dotted lines of the management scenarios reflect the intermediate period, while solid lines reflect the management period. Additionally, two vertical lines correspond to the start and end of the management period.

Be aware that potential catch intervals of more a year, e.g. biennial assessment so that the intermediate period spans two years, or management period spans two years, are equally split up into annual intervals.

Be aware of the fact that the catches represent intervals, where the length of the interval is indicated by dtc, e.g. with $dtc = 1, C(1990) = [1990, 1990[$. In the plot the catches (and vertical lines) correspond to the beginning of the catch interval. It might thus seem as if the time of the vertical lines and the management interval would not align.

Value

Nothing.

Examples

```
data(pol)
rep <- fit.spict(pol$albacore)
plot2(rep)
```

plotmm.priors *Plot priors of Meyer & Millar model*

Description

Plot priors of Meyer & Millar model

Usage

```
## S3 method for class 'priors'
plot(nm, priorsin, add = TRUE, ...)
```

Arguments

nm	Name of prior
priorsin	List of priors, typically inp$meyermillar$priors.
add	If TRUE add to current plot.
...	Additional arguments to plot.

Value

Nothing.

plotspict.bbmsy *Plot estimated B/Bmsy.*

Description

Plot estimated B/Bmsy.

Usage

```
plotspict.bbmsy(
  rep,
  logax = FALSE,
  main = "Relative biomass",
  ylim = NULL,
  plot.obs = TRUE,
  qlegend = TRUE,
  lineat = 1,
  xlab = "Time",
  stamp = get.version(),
  verbose = TRUE,
  CI = 0.95,
  plot.spinup = FALSE
)
```

Arguments

rep	A result report as generated by running fit.spict.
logax	Take log of y-axis? default: FALSE
main	Title of plot.
ylim	Limits for y-axis.
plot.obs	If TRUE observations are plotted.
qlegend	If TRUE legend explaining colours of observation data is plotted.
lineat	Draw horizontal line at this y-value.
xlab	Label of x-axis.
stamp	Stamp plot with this character string.
verbose	Should detailed outputs be provided (default: TRUE).
CI	Confidence intervals to be calculated, e.g. 0.9 for the 90% confidence intervals. By default (CI = 0.95), the 95% confidence intervals are estimated.
plot.spinup	If TRUE spinup period is plotted, if spinup was used (inp$nspinup > 0). Default: FALSE.

Details

Plots estimated B/Bmsy.

If no management scenarios are included in rep$man, the grey vertical line corresponds to the time of the last observation. If management scenarios are included in rep$man, the prediction and confidence intervals of the base scenario (rep) are omitted and instead the projections of the different management scenarios are drawn in different colours. Dotted lines of the management scenarios reflect the intermediate period, while solid lines reflect the management period. Additionally, two vertical lines correspond to the start and end of the management period.

Value

Nothing.

Examples

```
data(pol)
rep <- fit.spict(pol$albacore)
plotspict.bbmsy(rep)
```

`plotspict.biomass` *Plot estimated biomass.*

Description

Plot estimated biomass.

Usage

```
plotspict.biomass(
  rep,
  logax = FALSE,
  main = "Absolute biomass",
  ylim = NULL,
  plot.obs = TRUE,
  qlegend = TRUE,
  xlab = "Time",
  ylab = NULL,
  rel.axes = TRUE,
  rel.ci = TRUE,
  stamp = get.version(),
  verbose = TRUE,
  CI = 0.95,
  plot.spinup = FALSE
)
```

Arguments

`rep`	A result report as generated by running fit.spict.
`logax`	Take log of y-axis? default: FALSE
`main`	Title of plot.
`ylim`	Limits for y-axis.
`plot.obs`	If TRUE observations are plotted.
`qlegend`	If TRUE legend explaining colours of observation data is plotted.
`xlab`	Label of x-axis.
`ylab`	Label of y-axis.
`rel.axes`	Plot secondary y-axis contatning relative level of F.
`rel.ci`	Plot confidence interval for relative level of F.
`stamp`	Stamp plot with this character string.
`verbose`	Should detailed outputs be provided (default: TRUE).
`CI`	Confidence intervals to be calculated, e.g. 0.9 for the 90% confidence intervals. By default (CI = 0.95), the 95% confidence intervals are estimated.
`plot.spinup`	If TRUE spinup period is plotted, if spinup was used (inp$nspinup > 0). Default: FALSE.

Details

Plots estimated biomass, Bmsy with confidence limits.

If no management scenarios are included in `rep$man`, the grey vertical line corresponds to the time of the last observation. If management scenarios are included in `rep$man`, the prediction and confidence intervals of the base scenario (`rep`) are omitted and instead the projections of the different management scenarios are drawn in different colours. Dotted lines of the management scenarios reflect the intermediate period, while solid lines reflect the management period. Additionally, two vertical lines correspond to the start and end of the management period.

Value

Nothing.

Examples

```
data(pol)
rep <- fit.spict(pol$albacore)
plotspict.biomass(rep)
```

---

plotspict.catch *Plot observed catch and predictions.*

Description

Plot observed catch and predictions.

Usage

```
plotspict.catch(
  rep,
  main = "Catch",
  ylim = NULL,
  qlegend = TRUE,
  lcol = "blue",
  xlab = "Time",
  ylab = NULL,
  stamp = get.version(),
  verbose = TRUE,
  CI = 0.95
)
```

Arguments

rep	A result report as generated by running fit.spict.
main	Title of plot.
ylim	Limits for y-axis.
qlegend	If TRUE legend explaining colours of observation data is plotted.
lcol	Colour of prediction lines.
xlab	Label of x-axis.
ylab	Label of y-axis.
stamp	Stamp plot with this character string.
verbose	Should detailed outputs be provided (default: TRUE).
CI	Confidence intervals to be calculated, e.g. 0.9 for the 90% confidence intervals. By default (CI = 0.95), the 95% confidence intervals are estimated.

Details

Plots observed catch and predictions using the current F and Fmsy. The plot also contains the equilibrium catch if the current F is maintained. If no management scenarios are included in rep$man, the grey vertical line corresponds to the time of the last observation.

If management scenarios are included in rep$man, the prediction and confidence intervals of the base scenario (rep) are omitted and instead the projections of the different management scenarios are drawn in different colours. Generally, dotted lines of the management scenarios reflect the intermediate period, while solid lines reflect the management period. The catch of management period which are longer than 1 year are split up equally into annual intervals. Two vertical lines correspond to the start and end of the management period, respectively. However, there are special cases in which there is only one or no vertical line drawn, the catch trajectories are missing completely, or the line of the catch trajectory is solid even in the intermediate period. These cases and their implications on the annual catch plot are described in the following:

- If the management period is shorter than a year, no catch trajectories are drawn and there is only one vertical line indicating the start of the assessment period.
- If the management timeline differs between the scenarios in rep$man, no vertical lines are drawn as they would be at different times for each scenario.
- If the management period cannot be split equally into annual intervals, e.g. because it is 1.5 years long, the uneven remains are not displayed, in this example only the catch representative of one year is displayed. Additionally, the second vertical line indicating the end of the management period is omitted.
- If the intermediate period is shorter or longer than a year, e.g. 0.5 or 1.25 years, the lines of the management period start at the time of the last observation, because the catch in the intermediate period cannot be aggregated and displayed correctly. Additionally, the first vertical line indicating the start of the management period is omitted.

All catches in SPiCT represent intervals, where the length of the interval is indicated by dtc, e.g. with $dtc = 1, C(1990) = [1990, 1990[$. In the plot the catches (and vertical lines) correspond to the beginning of the catch interval. It might thus seem as if the time of the vertical lines and the management interval would not align.

Value

Nothing.

Examples

```
data(pol)
rep <- fit.spict(pol$albacore)
plotspict.catch(rep)
```

`plotspict.ci` *Plot catch and index data.*

Description

Plot catch and index data.

Usage

```
plotspict.ci(inp, stamp = get.version())
```

Arguments

`inp`	An input list containing data.
`stamp`	Stamp plot with this character string.

Value

Nothing

`plotspict.compare` *Compare different spict fits*

Description

Compare different spict fits

Usage

```
plotspict.compare(
  rep,
  ...,
  varname = c("B", "F", "C", "BBmsy", "FFmsy", "P"),
  exp = TRUE,
  CI = 0.95,
  plot.unc = TRUE,
  col = c("dodgerblue2", "darkorange1", "forestgreen", "goldenrod1", "purple2",
    "firebrick3", "skyblue4", "darkgreen", "salmon3", "brown2"),
  asp = 2,
  plot.legend = TRUE,
  stamp = get.version()
)
```

Arguments

rep	A result report as generated by running fit.spict or a list containing objects fitted with spict.
...	Optional additional spict fits.
varname	Name of the variable to be plotted. The following options are currently implemented: "B" for the biomass, "F" for the fishing mortality, "C" for the catch, "BBmsy" for the biomass, "FFmsy" for the fishing mortality, and "P" for the surplus production.
exp	Logical; indicating whether to plot the results on the natural or logarithmic scale. By default (TRUE), results are plotted in the natural scale.
CI	Confidence intervals to be calculated, e.g. 0.9 for the 90 confidence intervals. By default (0.95), the 95 intervals are estimated and plotted.
plot.unc	Logical or integer indicating whether to include the uncertainty intervals (CIs). If equal to TRUE (or 1), CIs are drawn as dashed lines, if equal to 2, CIs are drawn as shaded polygons. Default: TRUE.
col	Colours for different spict fits.
asp	Positive number deining the target aspect ratio (columns / rows) of the plot arrangement.
plot.legend	Logical; Indicating whether to include a legend. Default: TRUE. Use plot.legend = 2 to plot the legend inside the last panel.
stamp	Stamp plot with this character string.

Details

This function plots the results of spict fits in a single plot

Examples

```
data(pol)
inp <- pol$albacore
```

```
inp$dteuler <- 1/8
rep1 <- fit.spict(inp)
inp$priors$logbkfrac <- c(log(0.3), 0.1, 1)
rep2 <- fit.spict(inp)
plotspict.compare(list(def = rep1, bkprior = rep2))
```

`plotspict.compare.one` *Compare one variable of different spict fits*

Description

Compare one variable of different spict fits

Usage

```
plotspict.compare.one(
  rep,
  ...,
  varname = c("B", "F", "C", "BBmsy", "FFmsy", "P"),
  exp = TRUE,
  CI = 0.95,
  plot.unc = 1,
  col = c("dodgerblue2", "darkorange1", "forestgreen", "goldenrod1", "purple2",
    "firebrick3", "skyblue4", "darkgreen", "salmon3", "brown2"),
  xlab = NULL,
  ylab = NULL,
  main = NULL,
  plot.legend = TRUE,
  stamp = get.version()
)
```

Arguments

`rep`	A result report as generated by running `fit.spict` or a list containing objects fitted with spict.
`...`	Optional additional spict fits.
`varname`	Name of the variable to be plotted. The following options are currently implemented: `"B"` for the biomass, `"F"` for the fishing mortality, `"C"` for the catch, `"BBmsy"` for the biomass, `"FFmsy"` for the fishing mortality, and `"P"` for the surplus production.
`exp`	Logical; indicating whether to plot the results on the natural or logarithmic scale. By default (`TRUE`), results are plotted in the natural scale.
`CI`	Confidence intervals to be calculated, e.g. 0.9 for the 90 confidence intervals. By default (`0.95`), the 95 intervals are estimated and plotted.

plot.unc	Logical or integer indicating whether to include the uncertainty intervals (CIs). If equal to TRUE (or 1), CIs are drawn as dashed lines, if equal to 2, CIs are drawn as shaded polygons. Default: TRUE.
col	Colours for different spict fits.
xlab	Label for x-axis. By default (NULL), title is determined by variable (varname).
ylab	Label for y-axis. By default (NULL), title is determined by variable (varname).
main	Title of plot. By default (NULL), title is determined by variable (varname).
plot.legend	Logical; Indicating whether to include a legend. Default: TRUE.
stamp	Stamp plot with this character string.

Details

This function plots the results of the hindcasting cross validation analysis for each index.

Examples

```
data(pol)
inp <- pol$albacore
inp$dteuler <- 1/8
rep1 <- fit.spict(inp)
inp$priors$logbkfrac <- c(log(0.3), 0.1, 1)
rep2 <- fit.spict(inp)
## Not run:
plotspict.compare.one(rep1, rep2, varname = "F")

## End(Not run)
```

---

plotspict.data *Plot input data*

Description

Plot input data

Usage

```
plotspict.data(
  inpin,
  MSY = NULL,
  one.index = NULL,
  qlegend = TRUE,
  stamp = get.version()
)
```

Arguments

inpin	An input list containing data.
MSY	Value of MSY.
one.index	Integer indicating the number of the index to plot.
qlegend	If TRUE legend explaining colours of observation data is plotted.
stamp	Stamp plot with this character string.

Value

Nothing

---

plotspict.diagnostic *Plot model diagnostic (data, residuals, and more)*

Description

Plot model diagnostic (data, residuals, and more)

Usage

```
plotspict.diagnostic(
  rep,
  lag.max = 4,
  qlegend = TRUE,
  plot.data = TRUE,
  mfcol = FALSE,
  stamp = get.version()
)
```

Arguments

rep	A result report as generated by running fit.spict.
lag.max	Maximum lag to use in acf calculations.
qlegend	If TRUE plot a legend showing quarter of year information.
plot.data	If TRUE plot data in the top row (this option is only applied if osa residuals have been calculated).
mfcol	If TRUE plot plots columnwise (FALSE => rowwise).
stamp	Stamp plot with this character string.

Value

Nothing.

Examples

```
data(pol)
rep <- fit.spict(pol$albacore)
rep <- calc.osa.resid(rep)
plotspict.diagnostic(rep)
```

`plotspict.diagnostic.process` *Plot model diagnostics regarding processes and process residuals*

Description

Plot model diagnostics regarding processes and process residuals

Usage

```
plotspict.diagnostic.process(
  rep,
  lag.max = 4,
  qlegend = TRUE,
  plot.data = TRUE,
  mfcol = FALSE,
  add.loess = FALSE,
  span = 0.75,
  stamp = get.version()
)
```

Arguments

`rep`	A result report as generated by running `fit.spict` that contains process residuals calculated by the function `process.resid`.
`lag.max`	Maximum lag to use in acf calculations.
`qlegend`	If TRUE plot a legend showing quarter of year information.
`plot.data`	If TRUE plot data in the top row (this option is only applied if osa residuals have been calculated).
`mfcol`	If TRUE plot plots columnwise (FALSE => rowwise).
`add.loess`	Add smooth line (using `loess`) to residuals (Default: FALSE).
`span`	Parameter that controls the degree of smoothing (only used if `add.loess=TRUE`, Default: 0.75).
`stamp`	Stamp plot with this character string.

Value

Invisible `NULL`

See Also

process.resid

Examples

```
data(pol)
inp <- pol$albacore
rep <- fit.spict(inp)
rep <- calc.process.resid(rep)
plotspict.diagnostic.process(rep)
```

---

plotspict.f *Plot estimated fishing mortality.*

Description

Plot estimated fishing mortality.

Usage

```
plotspict.f(
  rep,
  logax = FALSE,
  main = "Absolute fishing mortality",
  ylim = NULL,
  plot.obs = TRUE,
  qlegend = TRUE,
  xlab = "Time",
  ylab = NULL,
  rel.axes = TRUE,
  rel.ci = TRUE,
  stamp = get.version(),
  verbose = TRUE,
  CI = 0.95,
  plot.spinup = FALSE
)
```

Arguments

rep	A result report as generated by running fit.spict.
logax	Take log of y-axis? default: FALSE
main	Title of plot.
ylim	Limits for y-axis.
plot.obs	If TRUE observations are plotted.
qlegend	If TRUE legend explaining colours of observation data is plotted.

xlab	Label of x-axis.
ylab	Label of y-axis.
rel.axes	Plot secondary y-axis contatning relative level of F.
rel.ci	Plot confidence interval for relative level of F.
stamp	Stamp plot with this character string.
verbose	Should detailed outputs be provided (default: TRUE).
CI	Confidence intervals to be calculated, e.g. 0.9 for the 90% confidence intervals. By default (CI = 0.95), the 95% confidence intervals are estimated.
plot.spinup	If TRUE spinup period is plotted, if spinup was used (inp$nspinup > 0). Default: FALSE.

Details

Plots estimated fishing mortality with Fmsy and associated confidence interval.

If no management scenarios are included in `rep$man`, the grey vertical line corresponds to the time of the last observation. If management scenarios are included in `rep$man`, the prediction and confidence intervals of the base scenario (`rep`) are omitted and instead the projections of the different management scenarios are drawn in different colours. Dotted lines of the management scenarios reflect the intermediate period, while solid lines reflect the management period. Additionally, two vertical lines correspond to the start and end of the management period.

Value

Nothing.

Examples

```
data(pol)
rep <- fit.spict(pol$albacore)
plotspict.f(rep)
```

---

plotspict.fb *Plot fishing mortality versus biomass.*

Description

Plot fishing mortality versus biomass.

Usage

```
plotspict.fb(
  rep,
  logax = FALSE,
  plot.legend = TRUE,
  man.legend = TRUE,
  ext = TRUE,
  rel.axes = FALSE,
  xlim = NULL,
  ylim = NULL,
  labpos = c(1, 1),
  xlabel = NULL,
  stamp = get.version(),
  verbose = TRUE,
  CI = 0.95
)
```

Arguments

rep	A result report as generated by running fit.spict.
logax	Take log of x and y-axes? default: FALSE
plot.legend	Plot legend explaining triangle.
man.legend	Plot legend explaining management scenarios..
ext	Add relative level axis to top and right side.
rel.axes	Plot axes in relative levels instead of absolute.
xlim	Limits of x-axis.
ylim	Limits of y-axis.
labpos	Positions of time stamps of start and end points as in pos in text().
xlabel	Label of x-axis. If NULL not used.
stamp	Stamp plot with this character string.
verbose	Should detailed outputs be provided (default: TRUE).
CI	Confidence intervals to be calculated, e.g. 0.9 for the 90% confidence intervals. By default (CI = 0.95), the 95% confidence intervals are estimated.

Details

Plots estimated fishing mortality as a function of biomass together with reference points and the prediction for next year given a constant F. The equilibrium biomass for F fixed to the current value is also plotted.

The predicted trajectory (or trajectories of different management scenarios) are only plotted for annnual data.

Value

Nothing.

Examples

```
data(pol)
rep <- fit.spict(pol$albacore)
plotspict.fb(rep)
```

`plotspict.ffmsy` *Plot estimated relative fishing mortality.*

Description

Plot estimated relative fishing mortality.

Usage

```
plotspict.ffmsy(
  rep,
  logax = FALSE,
  main = "Relative fishing mortality",
  ylim = NULL,
  plot.obs = TRUE,
  qlegend = TRUE,
  lineat = 1,
  xlab = "Time",
  stamp = get.version(),
  verbose = TRUE,
  CI = 0.95,
  plot.spinup = FALSE
)
```

Arguments

`rep`	A result report as generated by running fit.spict.
`logax`	Take log of y-axis? default: FALSE
`main`	Title of plot.
`ylim`	Limits for y-axis.
`plot.obs`	If TRUE observations are plotted.
`qlegend`	If TRUE legend explaining colours of observation data is plotted.
`lineat`	Draw horizontal line at this y-value.
`xlab`	Label of x-axis.
`stamp`	Stamp plot with this character string.
`verbose`	Should detailed outputs be provided (default: TRUE).
`CI`	Confidence intervals to be calculated, e.g. 0.9 for the 90% confidence intervals. By default (CI = 0.95), the 95% confidence intervals are estimated.
`plot.spinup`	If TRUE spinup period is plotted, if spinup was used (inp$nspinup > 0). Default: FALSE.

Details

Plots estimated fishing mortality with Fmsy and associated confidence interval.

If no management scenarios are included in `rep$man`, the grey vertical line corresponds to the time of the last observation. If management scenarios are included in `rep$man`, the prediction and confidence intervals of the base scenario (`rep`) are omitted and instead the projections of the different management scenarios are drawn in different colours. Dotted lines of the management scenarios reflect the intermediate period, while solid lines reflect the management period. Additionally, two vertical lines correspond to the start and end of the management period.

Value

Nothing.

Examples

```
data(pol)
rep <- fit.spict(pol$albacore)
plotspict.ffmsy(rep)
```

---

plotspict.growth *Plot estimated time-varying growth*

Description

Plot estimated time-varying growth

Usage

```
plotspict.growth(
  rep,
  logax = FALSE,
  main = "Time-varying growth",
  ylim = NULL,
  xlim = NULL,
  xlab = "Time",
  plot.ci = TRUE,
  stamp = get.version(),
  CI = 0.95
)
```

Arguments

`rep`	A result report as generated by running fit.spict.
`logax`	Take log of y-axis? default: FALSE
`main`	Title of plot.

ylim	Limits for y-axis.
xlim	Limits for x-axis.
xlab	Label of x-axis.
plot.ci	If TRUE 95% CIs are included.
stamp	Stamp plot with this character string.
CI	Confidence intervals to be calculated, e.g. 0.9 for the 90% confidence intervals. By default (CI = 0.95), the 95% confidence intervals are estimated.

Details

Plots estimated time-varying growth

Value

Nothing.

`plotspict.hcr` *Comparison plot of aavailable management scenarios*

Description

A comparison of all management scenarios in `rep$man`. For each scenario, the fishing mortality in the management period is shown against the relative biomass at the point of the harvest control rule (HCR) evaluation.

Usage

```
plotspict.hcr(rep, xlim = c(0, 3), CI = 0.95)
```

Arguments

rep	A result report as generated by running `fit.spict` that contains management scenarios added by `manage` or `add.man.scenario`.
xlim	Numeric vector with the lower and upper x-axis limits
CI	Confidence intervals to be calculated, e.g. 0.9 for the 90% confidence intervals. By default (CI = 0.95), the 95% confidence intervals are estimated.

Value

Invisible `NULL`

See Also

`manage`

Examples

```
rep <- fit.spict(pol$hake)
rep <- manage(rep)
plotspict.hcr(rep)
```

`plotspict.hindcast` *Hindcasting plot for indices*

Description

Hindcasting plot for indices

Usage

```
plotspict.hindcast(
  rep,
  add.mase = TRUE,
  CI = 0.95,
  verbose = TRUE,
  xlim = NULL,
  ylim = NULL,
  xlab = "Year",
  ylab = NA,
  plot.log = TRUE,
  mfrow = NULL,
  mar = c(2, 2, 3, 1) + 0.1,
  oma = c(3, 3, 1, 1),
  legend.title = NULL,
  legend.pos = "topright",
  legend.ncol = 1,
  asp = 2,
  stamp = get.version()
)
```

Arguments

`rep`	A result report as generated by running `fit.spict` that contains hindcasted runs added by `hindcast`.
`add.mase`	Calculate Mean Absolute Scaled Error (MASE) and add it to the plots.
`CI`	Confidence intervals to be calculated, e.g. 0.9 for the 90% confidence intervals. By default (CI = 0.95), the 95% confidence intervals are estimated.
`verbose`	Should detailed outputs be provided (default: TRUE).
`xlim`	Limits for x-axis.
`ylim`	Limits for y-axis.

xlab	Label for x-axis, "Year" by default.
ylab	Label for y-axis, "Index" by default.
plot.log	Should values be plotted on the log scale (default: TRUE).
mfrow	A vector of the form 'c(nr, nc)'. Subsequent figures will be drawn in an 'nr'-by-'nc' array on the device by _rows_. Default: NULL.
mar	A numerical vector of the form 'c(bottom, left, top, right)' which gives the number of lines of margin to be specified on the four sides of the plot. The default is 'c(2, 2, 3, 1) + 0.1'.
oma	A vector of the form 'c(bottom, left, top, right)' giving the size of the outer margins in lines of text.
legend.title	Legend title. By default no title is used: "".
legend.pos	Legend position (default: "topright"). If NULL or NA, no legend is plotted.
legend.ncol	Legend number of columns (default: 1).
asp	positive number; the target aspect ratio (columns / rows) in the graphical output. Default: 2.
stamp	Stamp plot with this character string.

Details

This function plots the results of the hindcasting cross validation analysis for each index.

Value

MASE or Invisible NULL

References

Carvalho, F., Winker, H., Courtney, D., Kapur, M., Kell, L., Cardinale, M., Schirripa, M., Kitakado, T., Yemane, D., Piner, K.R. Maunder, M.N., Taylor, I., Wetzel, C.R., Doering, K., Johnsonm, K.F., Methot, R. D. (2021). A cookbook for using model diagnostics in integrated stock assessments. Fisheries Research, 240, 105959.

Kell, L. T., Kimoto, A., & Kitakado, T. (2016). Evaluation of the prediction skill of stock assessment using hindcasting. Fisheries research, 183, 119-127.

Kell, L. T., Sharma, R., Kitakado, T., Winker, H., Mosqueira, I., Cardinale, M., & Fu, D. (2021). Validation of stock assessment methods: is it me or my model talking?. ICES Journal of Marine Science, 78(6), 2244-2255.

Winker, H., Carvalho, F., & Kapur, M. (2018). JABBA: just another Bayesian biomass assessment. Fisheries Research, 204, 275-288.

See Also

hindcast

Examples

```
data(pol)
inp <- pol$albacore
rep <- fit.spict(inp)
rep <- hindcast(rep, npeels = 5)
plotspict.hindcast(rep)
```

---

plotspict.infl *Plots influence statistics of observations.*

Description

Plots influence statistics of observations.

Usage

```
plotspict.infl(rep, stamp = get.version())
```

Arguments

rep	A valid result from calc.influence().
stamp	Stamp plot with this character string.

Details

TBA

Value

Nothing.

---

plotspict.inflsum *Plots summary of influence statistics of observations.*

Description

Plots summary of influence statistics of observations.

Usage

```
plotspict.inflsum(rep, stamp = get.version())
```

Arguments

rep	A valid result from calc.influence().
stamp	Stamp plot with this character string.

Details

TBA

Value

Nothing.

`plotspict.likprof` *Plots result of likelihood profiling.*

Description

Plots result of likelihood profiling.

Usage

```
plotspict.likprof(input, logpar = FALSE, stamp = get.version(), CI = 0.95)
```

Arguments

`input`	Result of running likprof.spict().
`logpar`	If TRUE log of parameters are shown.
`stamp`	Stamp plot with this character string.
`CI`	Confidence intervals to be calculated, e.g. 0.9 for the 90% confidence intervals. By default (CI = 0.95), the 95% confidence intervals are estimated.

Details

TBA

Value

Nothing but shows a plot.

plotspict.osar *Plot one-step-ahead residuals*

Description

Plot one-step-ahead residuals

Usage

```
plotspict.osar(rep, collapse.I = TRUE, qlegend = TRUE)
```

Arguments

rep	A result report as generated by running fit.spict.
collapse.I	Collapse index residuals into one plot. Default: TRUE.
qlegend	Plot legend for quarters.

Details

Plots observed versus predicted catches.

Value

Nothing.

Examples

```
data(pol)
rep <- fit.spict(pol$albacore)
rep <- calc.osa.resid(rep)
plotspict.osar(rep)
```

plotspict.priors *Plot priors and posterior distribution.*

Description

Plot priors and posterior distribution.

Usage

```
plotspict.priors(
  rep,
  do.plot = NULL,
  stamp = get.version(),
  automfrow = TRUE,
  CI = 0.95
)
```

Arguments

rep	A result from fit.spict.
do.plot	Optional integer defining maximum number of priors to plot. Set to NULL to plot all active priors. Default: NULL
stamp	Stamp plot with this character string.
automfrow	Automatically set 'mfrow' to see all priors in one plot? Not used if do.plot is set. Default: TRUE
CI	Confidence intervals to be calculated, e.g. 0.9 for the 90% confidence intervals. By default (CI = 0.95), the 95% confidence intervals are estimated.

Value

Nothing

---

plotspict.production *Plot theoretical production curve and estimates.*

Description

Plot theoretical production curve and estimates.

Usage

```
plotspict.production(
  rep,
  n.plotyears = 40,
  main = "Production curve",
  stamp = get.version(),
  CI = 0.95
)
```

Arguments

rep	A result report as generated by running fit.spict.
n.plotyears	Plot years next to points if number of points is below n.plotyears. Default: 40.
main	Title of plot.
stamp	Stamp plot with this character string.
CI	Confidence intervals to be calculated, e.g. 0.9 for the 90% confidence intervals. By default (CI = 0.95), the 95% confidence intervals are estimated.

Details

Plots the theoretical production curve (production as a function of biomass) as calculated from the estimated model parameters. Overlaid is the estimated production/biomass trajectory.

Value

Nothing.

Examples

```
data(pol)
rep <- fit.spict(pol$albacore)
plotspict.production(rep)
```

`plotspict.retro` *Plot results of retrospective analysis*

Description

Plot results of retrospective analysis

Usage

```
plotspict.retro(rep, stamp = get.version(), add.mohn = TRUE, CI = 0.95)

plotspict.retro.fixed(rep, CI = 0.95)
```

Arguments

`rep`	A valid result from fit.spict.
`stamp`	Stamp plot with this character string.
`add.mohn`	Adds Mohn's rho
`CI`	Confidence intervals to be calculated, e.g. 0.9 for the 90% confidence intervals. By default (CI = 0.95), the 95% confidence intervals are estimated.

Value

Ivisible `NULL`. If `add.mohn` is `TRUE`, `plotspict.retro` returns the Mohn's rho for B/Bmsy and F/Fmsy.

Note

The retrospective runs that did not converge are excluded from the plots and from the calculation of Mohn's rho. A message is displayed in such a case.

plotspict.season *Plot the mean F cycle*

Description

Plot the mean F cycle

Usage

```
plotspict.season(rep, stamp = get.version(), CI = 0.95)
```

Arguments

rep	A result report as generated by running fit.spict.
stamp	Stamp plot with this character string.
CI	Confidence intervals to be calculated, e.g. 0.9 for the 90% confidence intervals. By default (CI = 0.95), the 95% confidence intervals are estimated.

Details

If seasonal data are available the seasonal cycle in the fishing mortality can be estimated. This function plots this mean F cycle.

Value

Nothing.

plotspict.tc *Plot time constant.*

Description

Plot time constant.

Usage

```
plotspict.tc(rep, main = "Time to Bmsy", stamp = get.version(), CI = 0.95)
```

Arguments

rep	A result report as generated by running fit.spict.
main	Title of plot.
stamp	Stamp plot with this character string.
CI	Confidence intervals to be calculated, e.g. 0.9 for the 90% confidence intervals. By default (CI = 0.95), the 95% confidence intervals are estimated.

Details

Plots the time required for the biomass to reach a certain proportion of Bmsy. The time required to reach 95% of Bmsy is highlighted.

Value

Nothing.

Examples

```
data(pol)
rep <- fit.spict(pol$albacore)
plotspict.tc(rep)
```

---

pol *Fisheries data included in Polacheck et al. (1993).*

Description

Fisheries data included in Polacheck et al. (1993).

Usage

```
data(pol)
```

Format

Data are lists containing data and initial values for estimation formatted to be used as an input to fit.spict().

Details

Fisheries data for South Atlantic albacore, northern Namibian hake, and New Zealand rock lobster.

Source

Polacheck et al. (1993), Canadian Journal of Fisheries and Aquatic Science, vol 50, pp. 2597-2607.

Examples

```
data(pol)
rep <- fit.spict(inp=pol$albacore)
rep <- fit.spict(inp=pol$hake)
rep <- fit.spict(inp=pol$lobster)
```

predict.b *Helper function for sim.spict().*

Description

Helper function for sim.spict().

Usage

```
## S3 method for class 'b'
predict(B0, F0, gamma, m, K, n, dt, sdb, btype)
```

Arguments

B0	Initial biomass.
F0	Fishing mortality.
gamma	gamma parameter in Fletcher's Pella-Tomlinson formulation.
m	m parameter in Fletcher's Pella-Tomlinson formulation.
K	Carrying capacity.
n	Pella-Tomlinson exponent.
dt	Time step.
sdb	Standard deviation of biomass process.
btype	If 'lamperti' use Lamperti transformed equation, if 'naive' use naive formulation.

Value

Predicted biomass at the end of dt.

predict.logf *Helper function for sim.spict().*

Description

Helper function for sim.spict().

Usage

```
## S3 method for class 'logf'
predict(logF0, dt, sdf, efforttype)
```

Arguments

logF0	Fishing mortality.
dt	Time step.
sdf	Standard deviation of F process.
efforttype	If 1 use diffusion on logF, if 2 use diffusion of F with state dependent noise (this induces the drift term -0.5*sdf^2 in log domain)

Value

Predicted F at the end of dt.

---

predict.logmre *Helper function for sim.spict().*

Description

Helper function for sim.spict().

Usage

```
## S3 method for class 'logmre'
predict(logmre0, dt, sdm, psi, logm)
```

Arguments

logmre0	Initial value
dt	Time step.
sdm	Standard deviation of mre process.
psi	Degree of attraction toward mean.
logm	Mean logm.

Value

Predicted mre at the end of dt.

print.spictcls *Output a summary of a fit.spict() run.*

Description

Output a summary of a fit.spict() run.

Usage

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

Arguments

x	A result report as generated by running fit.spict.
...	additional arguments affecting the summary produced.

Value

Nothing.

probdev *Estimate deviation between targeted and realised probability of specified model variable hitting a specified reference level under given fishing mortality*

Description

Estimate deviation between targeted and realised probability of specified model variable hitting a specified reference level under given fishing mortality

Usage

```
probdev(
  ffac = 1,
  rep,
  var = "logBpBmsy",
  ref = 1,
  problevel = 0.95,
  reportmode = 1,
  getFrac = FALSE,
  verbose = FALSE
)
```

Arguments

ffac	Factor to multiply current fishing mortality by (default: 1)
rep	A result report as generated by running fit.spict.
var	A variable of the spict model (default: "logBpBmsy").
ref	Reference level relative to specified variable (default: 1)
problevel	Probability level of the risk aversion (default: 0.95).
reportmode	Integer between 0 and 2 determining which objects will be adreported (default: 1).
getFrac	logical; return realised fraction of relative state (default: FALSE).
verbose	logical; print realised fraction of relative state, fishing mortality factor, and deviation (default: FALSE).

Value

Returns deviation between targeted and realised probability of hitting specified reference levels under given fishing mortality

---

prune.baserun *Prune a fitted spict object to the core elements*

Description

Prune a fitted spict object to the core elements

Usage

```
prune.baserun(rep)
```

Arguments

rep	Result of fit.spict().

Value

Fitted spict core object.

put.ax *Adds the x-axis to influence plots*

Description

Adds the x-axis to influence plots

Usage

```
put.xax(rep)
```

Arguments

rep	A valid result from calc.influence().

Details

TBA

Value

Nothing.

read.aspic *Reads ASPIC input file.*

Description

Reads ASPIC input file.

Usage

```
read.aspic(filename)
```

Arguments

filename	Path of the ASPIC input file.

Details

Reads an input file following the ASPIC 7 format described in the ASPIC manual (found here http://www.mhprager.com/aspic.html).

Value

A list of input variables that can be used as input to fit.spict().

Examples

```
## Not run:
filename <- 'YFT-SSE.a7inp' # or some other ASPIC 7 input file
inp <- read.aspic(filename)
rep <- fit.spict(inp)
summary(rep)
plot(rep)

## End(Not run)
```

`read.aspic.res` *Reads the parameter estimates of an Aspic result file.*

Description

Reads the parameter estimates of an Aspic result file.

Usage

```
read.aspic.res(filename)
```

Arguments

`filename`	Name of the Aspic result file to read

Details

TBA

Value

Vector containing the parameter estimates.

`refpointci` *Draw CI around a reference point using polygon*

Description

Draw CI around a reference point using polygon

Usage

```
refpointci(t, ll, ul, cicol = "ivory2")
```

Arguments

t	Time vector.
ll	Lower limit.
ul	Upper limit.
cicol	Colour of polygon

Value

Spline design matrix.

---

res.diagn *Helper function for calc.osar.resid that calculates residual statistics.*

Description

Helper function for calc.osar.resid that calculates residual statistics.

Usage

```
res.diagn(resid, id, name = "")
```

Arguments

resid	Residuals from either catches or indices.
id	Identifier for residuals e.g. "C".
name	Identifier that will be used in warning messages.

Value

List containing residual statistics in 'diagn', shapiro output in 'shapiro', and bias output in 'bias'.

---

retape.spict *Retape a fitted spict model based on an updated input list*

Description

Retape a fitted spict model based on an updated input list

Usage

```
retape.spict(rep, inp, verbose = FALSE, dbg = 0, mancheck = TRUE)
```

Arguments

rep	A result report as generated by running fit.spict.
inp	Input list with updated settings
verbose	logical; print informative text (default: FALSE)
dbg	Debug flag, dbg=0 no output, dbg=1 some output, dbg=2 more output (default: 0).
mancheck	Logical; Should the time-dependent objects in inp be checked against the management time and correct if necessary? (Default: TRUE)

Details

This function reevaluates derived variables of a fitted spict model after updating model settings, such as the prediction horizon or fishing mortality during the predicted time period.

Value

Retaped spict model

---

retro *Conduct retrospective analysis*

Description

Conduct retrospective analysis

Usage

```
retro(rep, nretroyear = 5, reduce_output_size = TRUE, mc.cores = 1)
```

Arguments

rep	A valid result from fit.spict.
nretroyear	Number of years of data to remove (this is also the total number of model runs).
reduce_output_size	logical, if TRUE (default) the retro is run with getReportCovariance and getJointPrecision set as FALSE
mc.cores	Number of cores for parallel::mclapply function. By default 1.

Details

A retrospective analysis consists of estimating the model with later data points removed sequentially one year at a time.

Value

A rep list with the added key retro containing the results of the retrospective analysis. Use plotspict.retro() to plot these results.

Examples

```
data(pol)
inp <- pol$albacore
rep <- fit.spict(inp)
rep <- retro(rep, nretroyear=6)
plotspict.retro(rep)
```

---

season.cols *Load season colors.*

Description

Load season colors.

Usage

```
season.cols(modin)
```

Arguments

modin	Time vector modulo 1.

Value

Vector containing season colors.

---

shaperate2meanvar *Convert shape and rate of gamma distribution to mean and variance*

Description

Convert shape and rate of gamma distribution to mean and variance

Usage

```
shaperate2meanvar(shape, rate)
```

Arguments

shape	Shape parameter
rate	Rate parameter (scale = 1/rate).

Value

Vector containing mean and var parameters.

`shorten.inp` *Shorten time series of input data to specified range*

Description

Shorten time series of input data to specified range

Usage

```
shorten.inp(inp, mintime = NULL, maxtime = NULL)
```

Arguments

`inp`	An input list containing data.
`mintime`	Starting time. If `NULL` (default), keep from the start of the time series.
`maxtime`	Ending time. If `NULL` (default), keep until the end of the time series.

Details

Time is given in decimal notation (e.g. 2005.3). If both `start` and `end` are `NULL`, `inp` is returned after running `check.inp`.

If codemaxtime is not set, i.e. the first part of the time series is cut, then management settings are not changed. Otherwise default values are used (see `check.inp`). The `ir` vector, which sets up different regimes, is always overwritten.

Value

List of shortened input time series and input variables as it is returned by `check.inp`

Author(s)

T.K. Mildenberger <t.k.mildenberger@gmail.com>

See Also

`check.inp`

Examples

```
inp <- pol$albacore

## Keep only years from 1973 onwards
shorten.inp(inp, mintime = 1973)

## Keep years until 1985
shorten.inp(inp, maxtime = 1985)
## Not run:
## Empty data set gives an error
shorten.inp(inp, mintime = 1910, maxtime = 1930)

## End(Not run)
```

`sim.spict` *Simulate data from Pella-Tomlinson model*

Description

Simulate data from Pella-Tomlinson model

Usage

```
sim.spict(input, nobs = 100, use.tmb = FALSE, verbose = TRUE)
```

Arguments

`input`	Either an inp list with an ini key (see ?check.inp) or a rep list where rep is the output of running fit.spict().
`nobs`	Optional specification of the number of simulated observations.
`use.tmb`	Should the TMB functionality be used for simulation? (Default: FALSE)
`verbose`	Should detailed outputs be provided? (Default: TRUE)

Details

Simulates data using either manually specified parameters values or parameters estimated by fit.spict().

Manual specification: To specify parameters manually use the inp$ini format similar to when specifying initial values for running fit.spict(). Observations can be simulated at specific times using inp$timeC and inp$timeI. If these are not specified then the length of inp$obsC or inp$obsI is used to determine the number of observations of catches and indices respectively. If none of these are specified then nobs observations of catch and index will be simulated evenly distributed in time.

Estimated parameters: Simply take the output from a fit.spict() run and use as input to sim.spict().

Value

A list containing the simulated data.

Examples

```
data(pol)
repin <- fit.spict(pol$albacore)
# Simulate a specific number of observations
inp <- list()
inp$dteuler <- 1/4 # To reduce calculation time
inp$ini <- repin$inp$ini
inp$ini$logF <- NULL
inp$ini$logB <- NULL
set.seed(1)
sim <- sim.spict(inp, nobs=150)
repsim <- fit.spict(sim)
summary(repsim) # Note true values are listed in the summary
plot(repsim) # Note true states are shown with orange colour

# Simulate data with seasonal F
inp <- list()
inp$dteuler <- 1/4
inp$nseasons <- 2
inp$splineorder <- 1
inp$obsC <- 1:80
inp$obsI <- 1:80
inp$ini <- repin$inp$ini
inp$ini$logF <- NULL
inp$ini$logB <- NULL
inp$ini$logphi <- log(2) # Seasonality introduced here
inp <- check.inp(inp)
sim2 <- sim.spict(inp)
par(mfrow=c(2, 1))
plot(sim2$obsC, typ='l')
plot(sim2$obsI[[1]], typ='l')
```

`spict` *Stochastic surplus Production model in Continuous-Time (SPiCT)*

Description

Fits a continuous-time surplus production model to data.

Author(s)

Martin W. Pedersen <wpsgodd@gmail.com>

References

https://github.com/DTUAqua/spict

See Also

test.spict

Examples

```
rep <- test.spict()
```

`spictcls` *An S4 class to represent output from a SPiCT fit.*

Description

An S4 class to represent output from a SPiCT fit.

`summary.spictcls` *Output a summary of a fit.spict() run.*

Description

Output a summary of a fit.spict() run.

Usage

```
## S3 method for class 'spictcls'
summary(object, CI = 0.95, ...)
```

Arguments

`object`	A result report as generated by running fit.spict.
`CI`	Confidence intervals to be calculated, e.g. 0.9 for the 90% confidence intervals. By default (CI = 0.95), the 95% confidence intervals are estimated.
`...`	additional arguments affecting the summary produced.

Details

The output includes the parameter estimates with 95% confidence intervals, estimates of derived parameters (Bmsy, Fmsy, MSY) with 95% confidence intervals, and predictions of biomass, fishing mortality, and catch.

Value

Nothing. Prints a summary to the screen.

Examples

```
data(pol)
rep <- fit.spict(pol$albacore)
summary(rep)
```

sumspict.diagnostics *Diagnostics table*

Description

Diagnostics table

Usage

```
sumspict.diagnostics(rep, ndigits = 8)
```

Arguments

rep	A result report as generated by running fit.spict.
ndigits	Present values with this number of digits after the dot.

Value

data.frame containing diagnostics information.

sumspict.drefpoints *Deternistic reference points of a fit.spict() run.*

Description

Deternistic reference points of a fit.spict() run.

Usage

```
sumspict.drefpoints(rep, ndigits = 8, CI = 0.95)
```

Arguments

rep	A result report as generated by running fit.spict.
ndigits	Present values with this number of digits after the dot.
CI	Confidence intervals to be calculated, e.g. 0.9 for the 90% confidence intervals. By default (CI = 0.95), the 95% confidence intervals are estimated.

Value

data.frame containing deterministic reference points.

sumspict.fixedpars *Fixed parameters table.*

Description

Fixed parameters table.

Usage

```
sumspict.fixedpars(rep, ndigits = 8)
```

Arguments

rep	A result report as generated by running fit.spict.
ndigits	Present values with this number of digits after the dot.

Value

data.frame containing fixed parameter information.

sumspict.ini *Sensitivity to the initial parameter values*

Description

Sensitivity to the initial parameter values

Usage

```
sumspict.ini(rep, numdigits)
```

Arguments

rep	A result report as generated by running fit.spict.
numdigits	Present values with this number of digits after the dot.

Value

list containing diagnostics information.

`sumspict.manage` *Print management summary.*

Description

Print management summary.

Usage

```
sumspict.manage(
  rep,
  include.EBinf = FALSE,
  include.unc = FALSE,
  include.abs = FALSE,
  timeline = TRUE,
  verbose = TRUE
)

mansummary(
  repin,
  include.EBinf = FALSE,
  include.unc = FALSE,
  include.abs = FALSE,
  timeline = TRUE,
  verbose = TRUE
)
```

Arguments

`rep`, `repin`	A result object as generated by running `manage` or `add.man.scenario`.
`include.EBinf`	Include EBinf/Bmsy in the output.
`include.unc`	Include uncertainty of management quantities.
`include.abs`	Include absolute B and F in management summary.
`timeline`	(default: FALSE)
`verbose`	Should detailed outputs be provided (default: TRUE).

Value

List with data frame containing management summary ('est') and data frame containing uncertainty of management quantities ('unc') if `include.unc = TRUE`.

Examples

```
## Not run:
data(pol)
rep <- fit.spict(pol$albacore)
```

```
rep <- manage(rep, c(2,4,8))
sumspict.manage(rep)

## End(Not run)
```

`sumspict.parest` *Parameter estimates of a fit.spict() run.*

Description

Parameter estimates of a fit.spict() run.

Usage

```
sumspict.parest(rep, ndigits = 8, CI = 0.95)
```

Arguments

`rep`	A result report as generated by running fit.spict.
`ndigits`	Present values with this number of digits after the dot.
`CI`	Confidence intervals to be calculated, e.g. 0.9 for the 90% confidence intervals. By default (CI = 0.95), the 95% confidence intervals are estimated.

Value

data.frame containing parameter estimates.

`sumspict.predictions` *Predictions of a fit.spict() run.*

Description

Predictions of a fit.spict() run.

Usage

```
sumspict.predictions(rep, ndigits = 8, CI = 0.95)
```

Arguments

`rep`	A result report as generated by running fit.spict.
`ndigits`	Present values with this number of digits after the dot.

Value

data.frame containing predictions.

sumspict.priors *Fixed parameters table.*

Description

Fixed parameters table.

Usage

```
sumspict.priors(rep, ndigits = 8)
```

Arguments

rep	A result report as generated by running fit.spict.
ndigits	Present values with this number of digits after the dot.

Value

data.frame containing fixed parameter information.

sumspict.srefpoints *Stochastic reference points of a fit.spict() run.*

Description

Stochastic reference points of a fit.spict() run.

Usage

```
sumspict.srefpoints(rep, ndigits = 8, CI = 0.95)
```

Arguments

rep	A result report as generated by running fit.spict.
ndigits	Present values with this number of digits after the dot.
CI	Confidence intervals to be calculated, e.g. 0.9 for the 90% confidence intervals. By default (CI = 0.95), the 95% confidence intervals are estimated.

Value

data.frame containing stochastic reference points.

sumspict.states *State estimates of a fit.spict() run.*

Description

State estimates of a fit.spict() run.

Usage

```
sumspict.states(rep, ndigits = 8, CI = 0.95)
```

Arguments

rep	A result report as generated by running fit.spict.
ndigits	Present values with this number of digits after the dot.

Value

data.frame containing state estimates.

test.spict *Example of a spict analysis.*

Description

Example of a spict analysis.

Usage

```
test.spict(dataset = "albacore")
```

Arguments

dataset	Specify one of the three test data sets: 'albacore', 'hake', 'lobster'. These can be accessed with the command data(pol).

Details

Loads a data set, fits the model, calculates one-step-ahead residuals, plots the results.

Value

A result report as given by fit.spict().

Examples

```
rep <- test.spict()
```

trans2real *Get real parameter values from transformed ones.*

Description

Get real parameter values from transformed ones.

Usage

```
trans2real(vals, nms, chgnms = TRUE)
```

Arguments

vals	Parameters in transformed domain.
nms	Names of transformed parameters (including log etc.)
chgnms	Remove transformation indication from the parameter names (e.g. remove log from logK).

Value

Parameter values in the natural domain.

true.col *Load color of true values from simulation.*

Description

Load color of true values from simulation.

Usage

```
true.col()
```

Value

Color vector

txt.stamp *Add spict version to plot*

Description

Add spict version to plot

Usage

```
txt.stamp(string = get.version(), cex = 0.5, do.flag = NULL)
```

Arguments

string	Character string to stamp.
cex	Stamp cex.
do.flag	If NULL stamp will be added if not in a multi plot, i.e. mean(par()$mfrow) > 1

Value

Nothing

validate.spict *Simulate data and reestimate parameters*

Description

Simulate data and reestimate parameters

Usage

```
validate.spict(
  inp,
  nsim = 50,
  invec = c(15, 60, 240),
  estinp = NULL,
  backup = NULL,
  df.out = FALSE,
  summ.ex.file = NULL,
  type = "nobs",
  parnames = NULL,
  exp = NULL,
  mc.cores = 1,
  model = "spict"
)
```

Arguments

inp	An inp list with an ini key (see ?check.inp). If you want to use estimated parameters for the simulation create the inp$ini from the pl key of a result of fit.spict().
nsim	Number of simulated data sets in each batch.
invec	Vector containing the number of simulated observations of each data set in each batch.
estinp	The estimation uses the true parameters as starting guess. Other initial values to be used for estimation can be specified in estinp$ini.
backup	Since this procedure can be slow a filename can be specified in backup where the most recent results will be available.
df.out	Output data frame instead of list.
summ.ex.file	Save a summary example to this file (to check that parameters have correct priors or are fixed).
type	Specify what type of information is contained in invec. If type == 'nobs' then invec is assumed to be a vector containing the number of simulated observations of each data set in each batch. If type == 'logsdc' then invec is assumed to be a vector containing values of logsdc over which to loop.
parnames	Vector of parameter names to extract stats for.
exp	Should exp be taken of parameters?
mc.cores	Number of cores for `parallel::mclapply` function. By default 1.
model	If 'spict' estimate using SPiCT. If 'meyermillar' estimate using the model of Meyer & Millar (1999), this requires rjags and coda packages.

Details

Given input parameters simulate a number of data sets. Then estimate the parameters from the simulated data and compare with the true values. Specifically, the one-step-ahead residuals are checked for autocorrelation and the confidence intervals of the estimated Fmsy and Bmsy are checked for consistency.

WARNING: One should simulate at least 50 data sets and preferably more than 100 to obtain reliable results. This will take some time (potentially hours).

Value

A list containing the results of the validation with the following keys:

- "osarpvals" P-values of the Ljung-Box test for uncorrelated one-step-ahead residuals.
- "*msyci"Logical. TRUE if the true value of B/Fmsy was inside the 95% confidence interval for the estimate, otherwise FALSE
- "*msyciw" Width of the 95% confidence interval of the estimate of Bmsy/Fmsy.

Examples

```
data(pol)
rep0 <- fit.spict(pol$albacore)
inp <- list()
inp$ini <- rep0$pl
set.seed(1234)
validate.spict(inp, nsim=10, invec=c(30, 60), backup='validate.RData')
```

---

validation.data.frame *Collect results from the output of running validate.spict.*

Description

Collect results from the output of running validate.spict.

Usage

```
validation.data.frame(ss)
```

Arguments

ss	Output from validation.spict.

Value

A data frame containing the formatted validation results.

---

warning.stamp *Add warning sign to plot*

Description

Add warning sign to plot

Usage

```
warning.stamp()
```

Value

Nothing

write.aspic *Takes a SPiCT input list and writes it as an Aspic input file.*

Description

Takes a SPiCT input list and writes it as an Aspic input file.

Usage

```
write.aspic(input, filename = "spictout.a7inp", verbose = FALSE)
```

Arguments

input	List of input variables or the output of a simulation using sim.spict().
filename	Name of the file to write.
verbose	If true write information to screen.

Details

TBA

Value

Noting.

Examples

```
data(pol)
sim <- (pol$albacore)
write.aspic(sim)
```

write.bug.file *Write the BUGS code to a text file*

Description

Write the BUGS code to a text file

Usage

```
write.bug.file(priors, fn = "tmp.bug")
```

Arguments

priors	List of priors, typically coming from inp$meyermillar$priors.
fn	Filename of to put BUGS code in.

Details

The .bug file generated by this function contains code published in Meyer & Millar (1999).

Value

Nothing.

References

Meyer, R., & Millar, R. B. (1999). BUGS in Bayesian stock assessments. Canadian Journal of Fisheries and Aquatic Sciences, 56(6), 1078-1087.

Index