Package ‘spict’

October 28, 2015

Type Package

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

Version 0.2

Date 2015-10-09

Author Martin Waever Pedersen

Maintainer Martin Waever Pedersen <`mawp@dtu.dk`>

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

License GPL (>=3)

Depends R (>= 3.0),
TMB

LinkingTo TMB

Suggests ellipse,
parallel

LazyData true

R topics documented:

`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.col.legend` *Add a legend indicating point colors of quarters.*

Description

Add a legend indicating point colors of quarters.

Usage

```
add.col.legend(qs, cols, pch = 1)
```

Arguments

`qs`	Quarters to plot legend for.
`cols`	Vector containing colors.
`pch`	Point character.

Value

Nothing.

`annual` *Convert from quarterly (or other sub-annual) data to annual means or sums.*

Description

Convert from quarterly (or other sub-annual) data to annual means or sums.

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`	If type='mean' then annual mean is calculated, if type='sum' then annual sum is calculated.

Value

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

`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.EBinf` *Calculate E(Binfinity) the fished equilibrium.*

Description

Calculate E(Binfinity) 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.

Value

E(Binf).

`calc.gamma` *Calculate gamma from n*

Description

Calculate gamma from n

Usage

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

`calc.influence` *Calculates influence statistics of observations.*

Description

Calculates influence statistics of observations.

Usage

```
calc.influence(rep)
```

Arguments

`rep`	A valid result from fit.spict().

Details

TBA

Value

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

Examples

```
data(pol)
rep <- fit.spict(pol$albacore)
rep <- calc.osa.resid(rep) # This step is required
rep <- calc.influence(rep)
```

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

Description

Calculate one-step-ahead residuals.

Usage

```
calc.osa.resid(rep, dbg = 0)
```

Arguments

`rep`	A result report as generated by running fit.spict.
`dbg`	Does nothing. Only preserved for backward compatibility with old osar function.
`silent`	If true nothing will be printed.

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 in lag 1 using the Ljung-Box test (see Box.test).

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)
```

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

Description

Check list of input variables

Usage

```
check.inp(inp)
```

Arguments

`inp`	List of input variables, see details for required variables.

Details

Fills in defalut values if missing. Required inputs:

- "inp$obsC" Vector of catch observations.
- "inp$obsI" List containing vectors of index observations.
- "inpinilogr" Initial value(s) for logr (log intrinsic growth rate). Can be specified as a scalar, or a vector of length 2 or 4. If length(logr)=2 different r values are estimated for first and second half of the year, if length(logr)=4 different r values are estimated for the four quarters of the year.
- "inpinilogK" Initial value for logK (log carrying capacity).
- "inpinilogq" Initial value for logq (log catchability of index).
- "inpinilogsdb" Initial value for logsdb (log standard deviation of log biomass).
- "inpinilogsdf" Initial value for logsdf (log standard deviation of log fishing mortality).

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$dtc" Time interval for catches, e.g. for annual catches inp$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$nseasons" Number of within-year seasons in data. If inp$nseasons > 1 then a cyclic B spline is used to impose a seasonal pattern in F. The parameters of the spline are phi and are estimated. Valid values are 1, 2 or 4. Default: number of unique within-year time points present in data.

- Parameters

- "inpinilogn" Pella-Tomlinson exponent. Default: log(2) corresponding to the Schaefer formulation.
- "inpiniphi" Vector for cyclic B spline representing within-year seasonal variation. Default: rep(1, inp$nseasons).
- "inpinilogalpha" sdi = alpha*sdb. Default: log(1).
- "inpinilogbeta" sdc = beta*sdf. Default: log(1).
- "inpinilogF" Default: log(0.2*r).
- "inpinilogB" Default: log(0.5*K).

- Priors Priors on model parameters are assumed Gaussian and specified in a vector of length 3: c(log(mean), stdev in log, useflag). NOTE: if specifying a prior for logB, then a 4th 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 0. Example: inp$priors$logr <- c(log(0.8), 0.1, 1) Example: inp$priors$logB <- c(log(200), 0.2, 1, 1985) - Settings

- "inp$dtpredc" Length of catch prediction interval in years. Default: max(inp$dtc).
- "inp$timepredc" Predict catches in interval lengths given by $dtpredc until this time. Default: Time of last observation.
- "inp$timepredi" Predict index until this time. Default: Time of last observation.
- "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$robflagc" Flag indicating whether robust estimation should be used for catches (either 0 or 1). Default: 0.
- "inp$robflagi" Flag 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: min(inp$dtc).
- "inp$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).

Value

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

Examples

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

`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)
```

Arguments

`rep`	A result report as generated by running fit.spict.
`inp`	The input list used as input to the validation.spict function.

Details

TBA

Value

A list containing the relevant statistics.

Examples

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

`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, 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. 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.

Fixed effects:

- "logm" Log of maximum sustainable yield.
- "logK" Log of carrying capacity.
- "logq" Log of catchability vector.
- "logsdf" Log of standard deviation of fishing mortality process noise.
- "logsdb" Log of standard deviation of biomass process noise.

Optional parameters (which are normally not estimated):

- "phi" Used when inp$nseasons > 1 to specify the cyclic B spline representing seasonal variation.
- "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)
- "logn" Parameter determining the shape of the production curve as in the generalised form of Pella & Tomlinson (1969). Default: log(2).

Random effects:

- "logB" Log of the biomass process given by the continuous-time stochastic Schaefer 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: F_t = D_s*G_t, where D_s is a cyclic B spline and dG_t = sigma_F * dV, with dV being Brownian motion.

Derived parameters:

- "logr" Log of intrinsic growth rate (r = 4m/K).
- "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)
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`	

Value

AIC

`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.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.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)
```

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.)

Details

Helper function for extracting the value and uncertainty of a specific model parameter, random effect or derived quantity.

Value

A matrix with four columns containing respectively: 1) the lower 95

Examples

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

`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.

`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)
```

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.

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 <- 'logsdb'
inp$likprof$parrange <- c(log(0.05), log(0.4))
inp$likprof$nogridpoints <- 15
rep <- fit.spict(inp)
rep <- likprof.spict(rep)
plotspict.likprof(rep, logpar=TRUE)
```

`make.datin` *Create data list.*

Description

Create data list.

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.

`make.ellipse` *Calculate confidence ellipsis.*

Description

Calculate confidence ellipsis.

Usage

```
make.ellipse(inds, rep)
```

Arguments

`inds`	Indices of the two reported model parameters.
`rep`	A result report as generated by running fit.spict.

Details

Calculates the confidence ellipsis of two reported model parameters. This is particularly useful as a detailed view of the uncertainty of two correlated parameters.

Value

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

`make.obj` *Create TMB obj.*

Description

Create TMB obj.

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)
```

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`	The generated tex code will be stored in this file.
`summaryoutfile`	Output of the summary will be stored in this file as plain text.
`keep.figurefiles`	If TRUE generated figure files will not be cleaned up.
`keep.txtfiles`	If TRUE generated txt files 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.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.

`manage` *Calculate predictions under different management scenarios*

Description

Calculate predictions under different management scenarios

Usage

```
manage(repin, scenarios = "all", dbg = 0)
```

Arguments

`repin`	Result list from fit.spict().
`scenarios`	Vector of integers specifying which scenarios to run. Default: 'all'.
`dbg`	Debug flag, dbg=1 some output, dbg=2 more ourput.

Details

Scenarios that are currently implemented include:

- "1" Take a specific catch. Default catch: MSY.
- "2" Fish at Fmsy.
- "3" No fishing, reduce to 5% of last F.
- "4" Reduce F by X%. Default X = 25.
- "5" Increase F by X%. Default X = 25.

Value

List containing results of management calculations.

Examples

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

`mansummary` *Print management summary.*

Description

Print management summary.

Usage

```
mansummary(rep, ypred = 1)
```

Arguments

`rep`	Result list as output from manage().
`ypred`	Show results for ypred years into the future.

Value

Data frame containing management summary.

`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, ...)
```

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.
`...`	Additional plotting arguments.

Value

Nothing.

`plot.spictcls` *3x3 plot illustrating spict results.*

Description

3x3 plot illustrating spict results.

Usage

```
## S3 method for class 'spictcls'
plot(rep, logax = FALSE)
```

Arguments

`rep`	A result report as generated by running fit.spict.
`logax`	Take log of relevant axes? default: FALSE

Details

Create a 3x3 plot containing the following:

- 1. Biomass using plotspict.biomass().
- 2. One-step-ahead residuals, only if calculated, using plotspict.osar().
- 3. One-step-ahead auto-correlation function (only if calculated).
- 4. Estimated F versus estimated B using plotspict.fb().
- 5. Estimated fishing mortality using plotspict.f().
- 6. Observed versus predicted catches using plotspict.catch().
- 7. Observed versus theoretical production using plotspict.production().
- 8. Calculated time-constant using plotspict.tc().

Value

Nothing.

Examples

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

`plotspict.bbmsy` *Plot estimated B/Bmsy.*

Description

Plot estimated B/Bmsy.

Usage

```
plotspict.bbmsy(rep, logax = FALSE, main = -1, plot.legend = TRUE,
  ylim = NULL, plot.obs = TRUE, qlegend = TRUE)
```

Arguments

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

Details

Plots estimated B/Bmsy.

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 = -1, plot.legend = TRUE,
  ylim = NULL, plot.obs = TRUE, qlegend = TRUE)
```

Arguments

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

Details

Plots estimated biomass, Bmsy with confidence limits.

Value

Nothing.

Examples

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

`plotspict.btrend` *Plot the expected biomass trend*

Description

Plot the expected biomass trend

Usage

```
plotspict.btrend(rep)
```

Arguments

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

Value

Nothing.

`plotspict.catch` *Plot observed catch and predictions.*

Description

Plot observed catch and predictions.

Usage

```
plotspict.catch(rep, main = -1, plot.legend = TRUE, ylim = NULL,
  qlegend = TRUE, lcol = "blue")
```

Arguments

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

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.

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)
```

Arguments

`inp`	An input list containing data.

Value

Nothing

`plotspict.data` *Plot input data*

Description

Plot input data

Usage

```
plotspict.data(inpin, MSY = NULL, one.index = NULL)
```

Arguments

`MSY`	Value of MSY.
`one.index`	Integer indicating the number of the index to plot.
`inp`	An input list containing data.

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)
```

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).

Value

Nothing.

Examples

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

`plotspict.f` *Plot estimated fishing mortality.*

Description

Plot estimated fishing mortality.

Usage

```
plotspict.f(rep, logax = FALSE, main = -1, plot.legend = TRUE,
  ylim = NULL)
```

Arguments

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

Details

Plots estimated fishing mortality with Fmsy and associated confidence interval.

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, ext = TRUE,
  rel.axes = FALSE, xlim = NULL, ylim = NULL, labpos = c(1, 1))
```

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.
`ext`	Add relative level axis to top and right side.
`rel.axes`	Plot axes in relative levels instead of absolute.

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.

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 = -1, plot.legend = TRUE,
  ylim = NULL)
```

Arguments

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

Details

Plots estimated fishing mortality with Fmsy and associated confidence interval.

Value

Nothing.

Examples

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

`plotspict.infl` *Plots influence statistics of observations.*

Description

Plots influence statistics of observations.

Usage

```
plotspict.infl(rep)
```

Arguments

`rep`	A valid result from calc.influence().

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)
```

Arguments

`rep`	A valid result from calc.influence().

Details

TBA

Value

Nothing.

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

Description

Plots result of likelihood profiling.

Usage

```
plotspict.likprof(input, logpar = FALSE)
```

Arguments

`input`	Result of running likprof.spict().
`logpar`	If TRUE log of parameters are shown.

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 = 4)
```

Arguments

`rep`	A result from fit.spict.
`do.plot`	Integer defining maximum number of priors to plot.

Value

Nothing

`plotspict.prodrate` *Plot production rate as a function of biomass.*

Description

Plot production rate as a function of biomass.

Usage

```
plotspict.prodrate(rep)
```

Arguments

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

Details

OBSOLETE!

Value

Nothing.

Examples

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

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

Description

Plot theoretical production curve and estimates.

Usage

```
plotspict.production(rep, n.plotyears = 40)
```

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.

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)
```

Arguments

`rep`	A valid result from fit.spict.

Value

Nothing

`plotspict.season` *Plot the mean F cycle*

Description

Plot the mean F cycle

Usage

```
plotspict.season(rep)
```

Arguments

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

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)
```

Arguments

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

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)
```

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.

Value

Predicted biomass at the end of dt.

`prop.F` *Calculate management for changing F by a given factor.*

Description

Calculate management for changing F by a given factor.

Usage

```
prop.F(fac, inpin, repin, dbg = 0)
```

Arguments

`fac`	Factor to multiply current F with.
`inpin`	Input list.
`repin`	Results list.
`dbg`	Debug flag, dbg=1 some output, dbg=2 more ourput.

Value

List containing results of management calculations.

`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.

`retro` *Conduct retrospective analysis*

Description

Conduct retrospective analysis

Usage

```
retro(rep, nretroyear = 5)
```

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).

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()
```

Value

Vector containing season colors.

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

Description

Simulate data from Pella-Tomlinson model

Usage

```
sim.spict(input, nobs = 100)
```

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.

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)
# Simulate a specific number of observations
inp <- pol$albacore
inp$obsC <- NULL
inp$timeC <- NULL
inp$obsI <- NULL
inp$timeI <- NULL
set.seed(1)
sim <- sim.spict(inp, nobs=150)
repsim <- fit.spict(sim)
summary(repsim) # Note true values are listed in the summary
dev.new(width=10, height=10)
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 <- pol$albacore$ini
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` *Fits a continuous-time surplus production model to data*

Description

Fits a continuous-time surplus production model to data

Author(s)

Martin W. Pedersen <map@aqua.dtu.dk>

References

`https://github.com/martinwpedersen/spict/`

See Also

`test.spict`

Examples

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

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

Description

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

Usage

```
summary.spictcls(object, numdigits = 8)
```

Arguments

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

Details

The output includes, the convergence message from the optimiser, the likelihood value of the parameters, the parameter estimates with 95

Value

Nothing.

Examples

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

`take.c` *Calculate management when taking a constant catch (proxy for setting a TAC).*

Description

Calculate management when taking a constant catch (proxy for setting a TAC).

Usage

```
take.c(catch, inpin, repin, dbg = 0)
```

Arguments

`catch`	Annual catch to take in the prediction period.
`inpin`	Input list.
`repin`	Results list.
`dbg`	Debug flag, dbg=1 some output, dbg=2 more ourput.

Value

List containing results of management calculations.

`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()
```

`true.col` *Load color of true values.*

Description

Load color of true values.

Usage

```
true.col()
```

Value

Color vector

`validate.spict` *Simulate data and reestimate parameters*

Description

Simulate data and reestimate parameters

Usage

```
validate.spict(inp, nsim = 50, nobsvec = c(15, 60, 240), estinp = NULL,
  backup = NULL, df.out = FALSE)
```

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$ini from the pl key of a result of fit.spict().
`nsim`	Number of simulated data sets in each batch.
`nobsvec`	Vector containing the number of simulated observations of each data set in each batch.
`estinp`	The estimation uses the true parameters as starting guess. Other initial values to be used for estimation can be specified in estinp$ini.
`backup`	Since this procedure can be slow a filename can be specified in backup where the most recent results will be available.

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, nobsvec=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.

`validation.data.frame.old` *Collect results from the output of running validate.spict (this function is outdated).*

Description

Collect results from the output of running validate.spict (this function is outdated).

Usage

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

Arguments

`ss`	Output from validation.spict.

Value

A data frame containing the formatted validation results.

`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")
```

Arguments

`input`	List of input variables or the output of a simulation using sim.spict().
`filename`	Name of the file to write.

Details

TBA

Value

Noting.

Examples

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

Index