

# Package ‘spict’

August 9, 2016

**Type** Package

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

**Version** 1.0

**Date** 2016-06-09

**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)

**Copyright** Martin Waever Pedersen <mawp@dtu.dk>

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

**LinkingTo** TMB, RcppEigen

**Suggests** ellipse,  
parallel,  
mgcv,  
rjags,  
coda

**LazyData** true

**GithubRepo** spict

**GithubRef** HEAD

**GithubSHA1** dec25dd230b4e1fc7e98dde7842dcfb4252ecdc7

## R topics documented:

acf.signf . . . . .	4
add.catchunit . . . . .	4
add.col.legend . . . . .	5
add.col.legend.hor . . . . .	5
annual . . . . .	5
arrow.line . . . . .	6
calc.EBinf . . . . .	6

calc.gamma	7
calc.influence	7
calc.osa.resid	8
check.inp	8
extract.simstats	11
fd	12
fit.aspic	13
fit.jags	13
fit.meyermillar	14
fit.spict	15
get.AIC	17
get.colnms	17
get.cov	17
get.EBinf	18
get.msyvec	18
get.order	18
get.osar.pvals	19
get.par	19
get.spline	20
get.version	20
guess.m	21
invlogit	21
invlogp1	22
latex.figure	22
likprof.spict	23
list.possible.priors	24
make.datin	24
make.ellipse	24
make.obj	25
make.report	25
make.splinemat	26
manage	26
mansummary	27
meanvar2shaperate	28
plot.col	28
plot.spictcls	29
plotmm.priors	30
plotspict.bbmsy	30
plotspict.biomass	31
plotspict.btrend	32
plotspict.catch	32
plotspict.ci	33
plotspict.data	33
plotspict.diagnostic	34
plotspict.f	34
plotspict.fb	35
plotspict.ffmsy	36
plotspict.infl	37

plotspict.inflsum . . . . .	37
plotspict.likprof . . . . .	38
plotspict.osar . . . . .	38
plotspict.priors . . . . .	39
plotspict.production . . . . .	39
plotspict.retro . . . . .	40
plotspict.season . . . . .	40
plotspict.tc . . . . .	41
pol . . . . .	41
predict.b . . . . .	42
predict.logf . . . . .	43
print.spictcls . . . . .	43
prop.F . . . . .	44
put.ax . . . . .	44
read.aspic . . . . .	45
read.aspic.res . . . . .	45
refpointci . . . . .	46
res.diagn . . . . .	46
retro . . . . .	47
season.cols . . . . .	47
shaperate2meanvar . . . . .	48
sim.spict . . . . .	48
spict . . . . .	49
spictcls . . . . .	50
summary.spictcls . . . . .	50
sumspict.diagnostics . . . . .	51
sumspict.drefpoints . . . . .	51
sumspict.fixedpars . . . . .	51
sumspict.parest . . . . .	52
sumspict.predictions . . . . .	52
sumspict.priors . . . . .	52
sumspict.srefpoints . . . . .	53
sumspict.states . . . . .	53
take.c . . . . .	54
test.spict . . . . .	54
trans2real . . . . .	55
true.col . . . . .	55
txt.stamp . . . . .	55
validate.spict . . . . .	56
validation.data.frame . . . . .	57
warning.stamp . . . . .	57
write.aspic . . . . .	58
write.bug.file . . . . .	58

---

acf.signf	<i>Check whether ACF of residuals is significant in any lags.</i>
-----------	---

---

**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	<i>Add catch unit to label</i>
---------------	--------------------------------

---

**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	<i>Add a legend explaining colors of points (vertical orientation)</i>
----------------	--

---

**Usage**

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

**Value**

Nothing.

---

add.col.legend.hor	<i>Add a legend explaining colors of points (horizontal orientation)</i>
--------------------	--

---

**Usage**

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

**Value**

Nothing.

---

annual	<i>Convert from quarterly (or other sub-annual) data to annual means or sums.</i>
--------	---

---

**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	<i>Draw a line with arrow heads.</i>
------------	--------------------------------------

---

**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	<i>Calculate <math>E(Binfinity)</math>, i.e. the fished equilibrium.</i>
------------	--

---

**Usage**

```
calc.EBinf(K, n, F1, Fmsy, sdb2)
```

**Arguments**

K	The carrying capacity.
n	Pella-Tomlinson exponent.
F1	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.  $\text{Max}()$  is used to avoid negative values.

**Value**

$E(\text{Binf})$ .

---

calc.gamma	<i>Calculate gamma from n</i>
------------	-------------------------------

---

**Usage**

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

**Arguments**

n	Exponent of the Pella-Tomlinson surplus production equation.
---	--

---

calc.influence	<i>Calculates influence statistics of observations.</i>
----------------	---

---

**Usage**

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

**Arguments**

rep	A valid result from <code>fit.spict()</code> .
-----	--

**Details**

TBA

**Value**

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

---

calc.osa.resid	<i>Calculate one-step-ahead residuals.</i>
----------------	--

---

**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)
```

---

check.inp	<i>Check list of input variables</i>
-----------	--------------------------------------

---

**Usage**

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

**Arguments**

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



## 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  $\text{inp\$dtc}=1$ , for quarterly catches  $\text{inp\$dtc}=0.25$ . 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:  $\min(\text{diff}(\text{inp\$timeC}))$ .
- "inp\$dte" Time interval for effort observations. For annual effort  $\text{inp\$dte}=1$ , for quarterly effort  $\text{inp\$dte}=0.25$ . Default:  $\min(\text{diff}(\text{inp\$timeE}))$ .
- "inp\$nseasons" Number of within-year seasons in data. If  $\text{inp\$nseasons} > 1$  then a seasonal pattern is used in F. Valid values of  $\text{inp\$nseasons}$  are 1, 2 or 4. Default: number of unique within-year time points present in data.

- Initial parameter values

- "inp\$ini\$logn" Pella-Tomlinson exponent determining shape of production function. Default:  $\log(2)$  corresponding to the Schaefer formulation.
- "inp\$ini\$logm" Initial value for logm (log maximum sustainable yield). Default:  $\log(\text{mean}(\text{catch}))$ .
- "inp\$ini\$logK" Initial value for logK (log carrying capacity). Default:  $\log(4 \cdot \max(\text{catch}))$ .
- "inp\$ini\$logq" Initial value for logq (log catchability of index). Default:  $\log(\max(\text{index})/K)$ .
- "inp\$ini\$logsdb" Initial value for logsdb (log standard deviation of biomass process). Default:  $\log(0.2)$ .
- "inp\$ini\$logsdf" Initial value for logsdf (log standard deviation of fishing mortality process). Default:  $\log(0.2)$ .
- "inp\$ini\$logsdi" Initial value for logsdi (log standard deviation of index observation error). Default:  $\log(0.2)$ .
- "inp\$ini\$logsdci" Initial value for logsdci (log standard deviation of catch observation error). Default:  $\log(0.2)$ .
- "inp\$ini\$phi" Vector for cyclic B spline representing within-year seasonal variation. Default:  $\text{rep}(1, \text{inp\$nseasons})$ .
- "inp\$ini\$logsdu" Initial value for logsdu (log standard deviation of log U, the state of the coupled SDE representation of seasonality). Default:  $\log(0.1)$ .

- "inp\$ini\$loglambda" 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

- "inp\$ini\$logF" Log fishing mortality. Default: log(0.2\*r), with r derived from m and K.
- "inp\$ini\$logB" Log biomass. Default: log(0.5\*K).
- "inp\$ini\$logU" 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 0. 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 Example: Inverse gamma prior on sdb^2: inp\$priors\$isdb2gamma <- meanvar2shaperate(1/exp(inp\$ini\$logsdB)^2, 150^2)

- Settings/Options/Preferences

- "inp\$dtpredc" Length of catch prediction interval in years. Default: max(inp\$dtc). Should be 1 to get annual predictions and 0.25 for quarterly predictions.
- "inp\$timepredc" Predict catches in interval lengths given by \$dtpredc until this time. Default: Time of last observation. Example: inp\$timepredc <- 2012
- "inp\$timepredi" Predict index until this time. Default: Time of last observation. Example: inp\$timepredi <- 2012
- "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: 1/16 year.

- "inp\$phases" Phases can be used to fix/free parameters and estimate in different stages or phases. To fix e.g. log $r$  at inp\$ini\$log $r$  set inp\$phases\$log $r$  <- -1. To free log $\alpha$  and estimate in phase 1 set inp\$phases\$log $\alpha$  <- 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).

## 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.*

---

## 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	<i>Format date</i>
----	--------------------

---

**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	<i>Fits aspic to the data contained in the input file</i>
-----------	---

---

**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	<i>Fit the Meyer &amp; Millar model using rjags</i>
----------	---

---

**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`.

---

<code>fit.meyermillar</code>	<i>Fit the model of Meyer &amp; Millar (1999)</i>
------------------------------	---

---

**Usage**

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

**Arguments**

<code>mminp</code>	Input list similar to the input to <code>fit.spict()</code>
--------------------	---

**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**

```
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)
```

fit.spict

*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.

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  $\text{inp}\$nseasons > 1$  and  $\text{inp}\$seasontype = 2$ .
- "logsde" Log of standard deviation of observation error of effort data. Only used if effort data is part of input.
- "logl1robfac" 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  $\text{inp}\$robflag = 1$  and/or  $\text{inp}\$robflag1 = 1$ .
- "logittp" Logit of the proportion of narrow distribution when using a mixture distribution robust toward outliers, i.e. when either  $\text{inp}\$robflag = 1$  and/or  $\text{inp}\$robflag1 = 1$ .

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:  $sdi = \exp(\text{logalpha}) * sdb$ . (normally set to  $\text{logalpha}=0$ )
- "logbeta" Proportionality factor for the observation noise of the catches and the fishing mortality process noise:  $sdc = \exp(\text{logbeta}) * sdf$ . (this is often difficult to estimate and can result in divergence of the optimisation. Normally set to  $\text{logbeta}=0$ )
- "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	<i>Calculate AIC from a rep list.</i>
---------	---------------------------------------

---

**Usage**

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

**Arguments**

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

**Value**

AIC

---

---

get.colnms	<i>Get column names for data.frames.</i>
------------	--

---

**Usage**

```
get.colnms()
```

**Value**

Vector containing column names of data frames.

---

---

get.cov	<i>Get covariance matrix of two reported quantities not of fixed model parameters. Covariance of fixed model parameters can be found in rep\$cov.fixed.</i>
---------	---

---

**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	<i>Calculate <math>E(Binfinity)</math> the fished equilibrium.</i>
-----------	--

---

**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	<i>If multiple growth rates (<math>r</math>) are used (e.g. for a seasonal model), return specified reference point for all instances of <math>r</math>.</i>
------------	--

---

**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.order	<i>Get order of printed quantities.</i>
-----------	---

---

**Usage**

```
get.order()
```

**Value**

Vector containing indices of printed quantities.

---

get.osar.pvals	<i>Check whether ACF of catch and index residuals is significant in any lags.</i>
----------------	---

---

**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	<i>Extract parameters from a result report as generated by fit.spict.</i>
---------	---

---

**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	<i>Get the values of the seasonal spline for F.</i>
------------	---

---

**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	<i>Get version of spict including git sha1 version if available.</i>
-------------	--

---

**Usage**

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

**Arguments**

pkg	Name of package.
-----	------------------

**Value**

Package version

---

guess.m	<i>Use a simple linear regression to guess <math>m</math> (MSY).</i>
---------	--

---

**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	<i>Inverse logit transform.</i>
----------	---------------------------------

---

**Usage**

```
invlogit(a)
```

**Arguments**

a	Value to take inverse logit of.
---	---------------------------------

**Value**

Inverse logit.

---

invlogp1	<i>Inverse log "plus one" transform</i>
----------	---

---

**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	<i>Generate latex code for including a figure.</i>
--------------	--

---

**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	<i>Create profile likelihood</i>
---------------	----------------------------------

---

**Usage**

```
likprof.spict(input, verbose = FALSE)
```

**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.

**Details**

The "likprof" list must contain 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*

---

**Usage**

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

**Value**

Prints parameters to which priors can be added.

---



---

```
make.datin
```

*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.ellipse
```

*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 using <code>TMB::MakeADFun</code> and <code>squelch</code> 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 <code>check.inp</code> .
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
-------------	---

---

**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 <code>fit.spic</code> t 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.
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.splinemat	<i>Make a spline design matrix</i>
----------------	------------------------------------

---

**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	<i>Calculate predictions under different management scenarios</i>
--------	---

---

**Usage**

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

**Arguments**

repin	Result list from fit.spict().
scenarios	Vector of integers specifying which scenarios to run. Default: 'all'.
manstart	Year that management should be initiated.
dbg	Debug flag, dbg=1 some output, dbg=2 more output.

## Details

Scenarios that are currently implemented include:

- "1" Keep the catch of the current year (i.e. the last observed catch).
- "2" Keep the F of the current year.
- "3" Fish at Fmsy i.e.  $F=F_{msy}$ .
- "4" No fishing, reduce to 1% of current F.
- "5" Reduce F by X%. Default X = 25.
- "6" 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)
```

---

<code>mansummary</code>	<i>Print management summary.</i>
-------------------------	----------------------------------

---

## Usage

```
mansummary(rep, ypred = 1, include.EBinf = FALSE, verbose = TRUE)
```

## Arguments

<code>rep</code>	Result list as output from <code>manage()</code> .
<code>ypred</code>	Show results for <code>ypred</code> years into the future.
<code>include.EBinf</code>	Include EBinf/Bmsy in the output.
<code>verbose</code>	Print more details on observed and predicted time intervals.

## Value

Data frame containing management summary.

---

meanvar2shaperate	<i>Convert mean and variance to shape and rate of gamma distribution</i>
-------------------	--

---

**Usage**

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

**Arguments**

mean	Mean value.
var	Variance.

**Value**

Vector containing shape and rate parameters.

---

plot.col	<i>Plot model points colored depending on the quarter to which they belong.</i>
----------	---

---

**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	<i>Plot summarising spict results.</i>
---------------	--

---

**Usage**

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

**Arguments**

x	A result report as generated by running fit.spict.
...	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().

**Value**

Nothing.

**Examples**

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

---

plotmm.priors	<i>Plot priors of Meyer &amp; Millar model</i>
---------------	--

---

**Usage**

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

**Arguments**

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

**Value**

Nothing.

---

plotspict.bbmsy	<i>Plot estimated B/Bmsy.</i>
-----------------	-------------------------------

---

**Usage**

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

**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.

**Details**

Plots estimated B/Bmsy.

**Value**

Nothing.

**Examples**

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

---

plotspict.biomass	<i>Plot estimated biomass.</i>
-------------------	--------------------------------

---

**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())
```

**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 containing relative level of F.
rel.ci	Plot confidence interval for relative level of F.
stamp	Stamp plot with this character string.

**Details**

Plots estimated biomass, Bmsy with confidence limits.

**Value**

Nothing.

**Examples**

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

---

plotspict.btrend	<i>Plot the expected biomass trend</i>
------------------	--

---

**Usage**

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

**Arguments**

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

**Value**

Nothing.

---

plotspict.catch	<i>Plot observed catch and predictions.</i>
-----------------	---

---

**Usage**

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

**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.

**Details**

Plots observed catch and predictions using the current  $F$  and  $F_{msy}$ . 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	<i>Plot catch and index data.</i>
--------------	-----------------------------------

---

**Usage**

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

**Arguments**

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

**Value**

Nothing

---

plotspict.data	<i>Plot input data</i>
----------------	------------------------

---

**Usage**

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

---

### Usage

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

### Arguments

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

**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 containing relative level of F.
rel.ci	Plot confidence interval for relative level of F.
stamp	Stamp plot with this character string.

**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	<i>Plot fishing mortality versus biomass.</i>
--------------	---

---

**Usage**

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

**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.

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.

### 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	<i>Plot estimated relative fishing mortality.</i>
-----------------	---

---

### Usage

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

### 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.

**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	<i>Plots influence statistics of observations.</i>
----------------	--

---

**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	<i>Plots summary of influence statistics of observations.</i>
-------------------	---

---

**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	<i>Plots result of likelihood profiling.</i>
-------------------	--

---

**Usage**

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

**Arguments**

input	Result of running likprof.spict().
logpar	If TRUE log of parameters are shown.
stamp	Stamp plot with this character string.

**Details**

TBA

**Value**

Nothing but shows a plot.

---

plotspict.osar	<i>Plot one-step-ahead residuals</i>
----------------	--------------------------------------

---

**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	<i>Plot priors and posterior distribution.</i>
------------------	--

---

**Usage**

```
plotspict.priors(rep, do.plot = 4, stamp = get.version())
```

**Arguments**

rep	A result from fit.spict.
do.plot	Integer defining maximum number of priors to plot.
stamp	Stamp plot with this character string.

**Value**

Nothing

---

plotspict.production	<i>Plot theoretical production curve and estimates.</i>
----------------------	---

---

**Usage**

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

**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.

**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	<i>Plot results of retrospective analysis</i>
-----------------	---

---

**Usage**

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

**Arguments**

rep	A valid result from fit.spict.
stamp	Stamp plot with this character string.

**Value**

Nothing

---

plotspict.season	<i>Plot the mean F cycle</i>
------------------	------------------------------

---

**Usage**

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

**Arguments**

rep	A result report as generated by running fit.spict.
stamp	Stamp plot with this character string.

**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	<i>Plot time constant.</i>
--------------	----------------------------

---

**Usage**

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

**Arguments**

rep	A result report as generated by running fit.spict.
main	Title of plot.
stamp	Stamp plot with this character string.

**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	<i>Fisheries data included in Polacheck et al. (1993).</i>
-----	--

---

**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$albacore)
rep <- fit.spict(inp=pol$shake)
rep <- fit.spict(inp=pol$lobster)
```

---

predict.b	<i>Helper function for sim.spict().</i>
-----------	---

---

**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	<i>Helper function for sim.spict().</i>
--------------	---

---

**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.

---

print.spictcls	<i>Output a summary of a fit.spict() run.</i>
----------------	---

---

**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.

---

prop.F	<i>Calculate management for changing F by a given factor.</i>
--------	---

---

**Usage**

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

**Arguments**

fac	Factor to multiply current F with.
inpin	Input list.
repin	Results list.
maninds	Indices of time vector for which to apply management.
corF	Make correction to F process such that the drift $(-0.5*sdf^2*dt)$ is cancelled and F remains constant in projection mode
dbg	Debug flag, dbg=1 some output, dbg=2 more output.

**Value**

List containing results of management calculations.

---

put.ax	<i>Adds the x-axis to influence plots</i>
--------	---

---

**Usage**

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

**Arguments**

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

**Details**

TBA

**Value**

Nothing.

---

read.aspic	<i>Reads ASPIC input file.</i>
------------	--------------------------------

---

**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	<i>Reads the parameter estimates of an Aspic result file.</i>
----------------	---

---

**Usage**

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

**Arguments**

filename	Name of the Aspic result file to read
----------	---------------------------------------

**Details**

TBA

**Value**

Vector containing the parameter estimates.

---

refpointci	<i>Draw CI around a reference point using polygon</i>
------------	---

---

**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	<i>Helper function for calc.osar.resid that calculates residual statistics.</i>
-----------	---

---

**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'.

---

retro	<i>Conduct retrospective analysis</i>
-------	---------------------------------------

---

**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 plot-spict.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	<i>Load season colors.</i>
-------------	----------------------------

---

**Usage**

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

**Arguments**

modin	Time vector modulo 1.
-------	-----------------------

**Value**

Vector containing season colors.

---

shaperate2meanvar	<i>Convert shape and rate of gamma distribution to mean and variance</i>
-------------------	--

---

**Usage**

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

**Arguments**

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

**Value**

Vector containing mean and var parameters.

---

sim.spict	<i>Simulate data from Pella-Tomlinson model</i>
-----------	---

---

**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)
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

*Fits a continuous-time surplus production model to data*


---

## Author(s)

Martin W. Pedersen <mawp@dtu.dk>

## References

<https://github.com/mawp/spict/>

## See Also

[test.spict](#)

## Examples

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

---

spictcls	<i>An S4 class to represent output from a SPiCT fit.</i>
----------	--

---

## Description

An S4 class to represent output from a SPiCT fit.

---

summary.spictcls	<i>Output a summary of a fit.spict() run.</i>
------------------	---

---

## Usage

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

## Arguments

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

## Details

The output includes the parameter estimates with 95

## Value

Nothing. Prints a summary to the screen.

## Examples

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

---

sumspict.diagnostics	<i>Diagnostics table</i>
----------------------	--------------------------

---

**Usage**

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

**Arguments**

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

**Value**

data.frame containing diagnostics information.

---

sumspict.drefpoints	<i>Deterministic reference points of a fit.spict() run.</i>
---------------------	---

---

**Usage**

```
sumspict.drefpoints(rep, numdigits = 8)
```

**Arguments**

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

**Value**

data.frame containing deterministic reference points.

---

sumspict.fixedpars	<i>Fixed parameters table.</i>
--------------------	--------------------------------

---

**Usage**

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

**Arguments**

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

**Value**

data.frame containing fixed parameter information.

---

sumspict.parest	<i>Parameter estimates of a fit.spict() run.</i>
-----------------	--

---

**Usage**

```
sumspict.parest(rep, numdigits = 8)
```

**Arguments**

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

**Value**

data.frame containing parameter estimates.

---

sumspict.predictions	<i>Predictions of a fit.spict() run.</i>
----------------------	--

---

**Usage**

```
sumspict.predictions(rep, numdigits = 8)
```

**Arguments**

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

**Value**

data.frame containing predictions.

---

sumspict.priors	<i>Fixed parameters table.</i>
-----------------	--------------------------------

---

**Usage**

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

**Arguments**

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

**Value**

data.frame containing fixed parameter information.

---

sumspict.srefpoints	<i>Stochastic reference points of a fit.spict() run.</i>
---------------------	--

---

**Usage**

```
sumspict.srefpoints(rep, numdigits = 8)
```

**Arguments**

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

**Value**

data.frame containing stochastic reference points.

---

sumspict.states	<i>State estimates of a fit.spict() run.</i>
-----------------	--

---

**Usage**

```
sumspict.states(rep, numdigits = 8)
```

**Arguments**

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

**Value**

data.frame containing state estimates.

---

take.c	<i>Calculate management when taking a constant catch (proxy for setting a TAC).</i>
--------	---

---

**Usage**

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

**Arguments**

catch	Annual catch to take in the prediction period.
inpin	Input list.
repin	Results list.
maninds	Indices of time vector for which to apply management.
dbg	Debug flag, dbg=1 some output, dbg=2 more output.

**Value**

List containing results of management calculations.

---

test.spict	<i>Example of a spict analysis.</i>
------------	-------------------------------------

---

**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	<i>Get real parameter values from transformed ones.</i>
------------	---

---

**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	<i>Load color of true values from simulation.</i>
----------	---

---

**Usage**

```
true.col()
```

**Value**

Color vector

---

txt.stamp	<i>Add spict version to plot</i>
-----------	----------------------------------

---

**Usage**

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

**Arguments**

string	Character string to stamp.
cex	Stamp cex.

**Value**

Nothing

---

validate.spict	<i>Simulate data and reestimate parameters</i>
----------------	--

---

## 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 = 8, 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 == 'logsd' then invec is assumed to be a vector containing values of logsd 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 to use.
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.*

---

**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*

---

**Usage**

```
warning.stamp()
```

**Value**

Nothing

---

write.aspic	<i>Takes a SPiCT input list and writes it as an Aspic input file.</i>
-------------	---

---

**Usage**

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

**Arguments**

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

**Details**

TBA

**Value**

Nothing.

**Examples**

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

---

write.bug.file	<i>Write the BUGS code to a text file</i>
----------------	---

---

**Usage**

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

**Arguments**

priors	List of priors, typically coming from <code>inp\$meyermillar\$priors</code> .
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

- \*Topic **assessment**
  - spict, [49](#)
- \*Topic **datasets**
  - pol, [41](#)
- \*Topic **fisheries**,
  - spict, [49](#)
- \*Topic **model**,
  - spict, [49](#)
- \*Topic **production**
  - spict, [49](#)
  
- acf.signf, [4](#)
- add.catchunit, [4](#)
- add.col.legend, [5](#)
- add.col.legend.hor, [5](#)
- annual, [5](#)
- arrow.line, [6](#)
  
- calc.EBinf, [6](#)
- calc.gamma, [7](#)
- calc.influence, [7](#)
- calc.osa.resid, [8](#)
- check.inp, [8](#)
  
- extract.simstats, [11](#)
  
- fd, [12](#)
- fit.aspic, [13](#)
- fit.jags, [13](#)
- fit.meyermillar, [14](#)
- fit.spict, [15](#)
  
- get.AIC, [17](#)
- get.colnms, [17](#)
- get.cov, [17](#)
- get.EBinf, [18](#)
- get.msyvec, [18](#)
- get.order, [18](#)
- get.osar.pvals, [19](#)
- get.par, [19](#)
- get.spline, [20](#)
  
- get.version, [20](#)
- guess.m, [21](#)
  
- invlogit, [21](#)
- invlogp1, [22](#)
  
- latex.figure, [22](#)
- likprof.spict, [23](#)
- list.possible.priors, [24](#)
  
- make.datin, [24](#)
- make.ellipse, [24](#)
- make.obj, [25](#)
- make.report, [25](#)
- make.splinemat, [26](#)
- manage, [26](#)
- mansummary, [27](#)
- meanvar2shaperate, [28](#)
  
- plot.col, [28](#)
- plot.priors (plotmm.priors), [30](#)
- plot.spictcls, [29](#)
- plotmm.priors, [30](#)
- plotspict.bbmsy, [30](#)
- plotspict.biomass, [31](#)
- plotspict.btrend, [32](#)
- plotspict.catch, [32](#)
- plotspict.ci, [33](#)
- plotspict.data, [33](#)
- plotspict.diagnostic, [34](#)
- plotspict.f, [34](#)
- plotspict.fb, [35](#)
- plotspict.ffmsy, [36](#)
- plotspict.infl, [37](#)
- plotspict.inflsum, [37](#)
- plotspict.likprof, [38](#)
- plotspict.osar, [38](#)
- plotspict.priors, [39](#)
- plotspict.production, [39](#)
- plotspict.retro, [40](#)

plotspict.season, 40  
plotspict.tc, 41  
pol, 41  
predict.b, 42  
predict.logf, 43  
print.spictcls, 43  
prop.F, 44  
put.ax, 44  
put.xax (put.ax), 44  
  
read.aspic, 45  
read.aspic.res, 45  
refpointci, 46  
res.diagn, 46  
retro, 47  
  
season.cols, 47  
shaperate2meanvar, 48  
sim.spict, 48  
spict, 49  
spict-package (spict), 49  
spictcls, 50  
spictcls-class (spictcls), 50  
summary.spictcls, 50  
sumspict.diagnostics, 51  
sumspict.drefpoints, 51  
sumspict.fixedpars, 51  
sumspict.parest, 52  
sumspict.predictions, 52  
sumspict.priors, 52  
sumspict.srefpoints, 53  
sumspict.states, 53  
  
take.c, 54  
test.spict, 49, 54  
trans2real, 55  
true.col, 55  
txt.stamp, 55  
  
validate.spict, 56  
validation.data.frame, 57  
  
warning.stamp, 57  
write.aspic, 58  
write.bug.file, 58