Package 'multiSA'

May 9, 2026

Title Multi-Stock Assessment

Version 0.1.1

Date 2026-03-20

Maintainer Quang Huynh <quang@bluematterscience.com>

Description

Implementation of a next-generation, multi-stock age-structured fisheries assessment model. 'multiSA' is intended for use in mixed fisheries where stock composition can not be readily identified in
fishery data alone, e.g., from catch and age/length composition. Models can be fitted to genetic data,
e.g., stock composition of catches and close-
kin pairs, with seasonal stock availability and movement.

License GPL (>= 3)

Depends R (>= 3.5.0)

Imports dplyr, gplots, graphics, methods, pbapply, reshape2,
rmarkdown, snowfall, stats, tinyplot, RTMB (>= 1.9), TMB, utils

Suggests numDeriv, RTMBdist, usethis

LazyLoad yes

Encoding UTF-8

RoxygenNote 7.3.3

URL https://blue-matter.github.io/multiSA/,
https://github.com/Blue-Matter/multiSA

BugReports https://github.com/Blue-Matter/multiSA/issues

NeedsCompilation no

Author Quang Huynh [aut, cre] (ORCID: <https://orcid.org/0000-0001-7835-4376>)

Repository CRAN

Date/Publication 2026-03-21 00:40:13 UTC

Contents

calc_eqdist . . . 3
calc_F . . . 3
calc_growth . . . 6
calc_index . . . 7
calc_LAK . . . 8
calc_nextN . . . 9
calc_phi_project . . . 10
calc_phi_simple . . . 11
calc_POP . . . 12
calc_population . . . 14
calc_recruitment . . . 16
check_data . . . 17
CondExpLt . . . 17
conv_mov . . . 19
conv_selpar . . . 20
conv_Sigma . . . 22
DCKMR-class . . . 23
Dfishery-class . . . 23
Dlabel-class . . . 25
Dmodel-class . . . 26
Dstock-class . . . 27
Dsurvey-class . . . 29
Dtag-class . . . 30
fit_MSA . . . 31
get_MSAdata . . . 32
get_sdreport . . . 33
like_CKMR . . . 34
like_comp . . . 34
make_parameters . . . 36
MSAassess-class . . . 39
MSAdata-class . . . 39
plot-MSA-data . . . 45
plot-MSA-state . . . 46
posfun . . . 48
prior . . . 49
profile . . . 50
report . . . 51
residuals . . . 52
retrospective . . . 53
simulate . . . 54
softmax . . . 55
summary . . . 56

Index **57**

calc_eqdist *Equilibrium distribution from movement matrix*

Description

Applies the seasonal movement matrices several times in order to obtain the equilibrium spatial distribution over the course of a year. Not used in the model but useful for reporting.

Usage

```
calc_eqdist(
  x,
  nm = dim(x)[1],
  nr = dim(x)[2],
  start = rep(1/nr, nr),
  m_start = 1L,
  nit = 20
)
```

Arguments

x	Movement array [m, r, r]. The second dimension corresponds to origin (sum to 1), and third dimension corresponds to destination
nm	Number of seasons
nr	Number of regions
start	The initial distribution. Vector of length nr
m_start	Integer, the season in which to apply the initial distribution and start the projection
nit	Integer, the number of times the movement matrix will be applied

Value

Matrix by season and region [m, r]

calc_F *Newton-Raphson search for fishing mortality*

Description

Performs a root finding routine to find the index of F that minimizes the difference between observed catch and the value predicted by the Baranov equation.

Usage

```
calc_F(
  Cobs,
  N,
  sel,
  wt,
  M,
  q_fs,
  delta = 1,
  na = dim(N)[1],
  nr = dim(N)[2],
  ns = dim(N)[3],
  nf = length(Cobs),
  Fmax = 2,
  nitF = 5L,
  trans = c("log", "logit")
)
```

Arguments

`Cobs`	Observed catch. Matrix `[f, r]`
`N`	Stock abundance at the beginning of the time step. Array `[a, r, s]`
`sel`	Selectivity. Array `[a, f, s]`
`wt`	Fishery weight at age. Array `[a, f, s]`
`M`	Instantaneous natural mortality. Units of per year `[a, s]`
`q_fs`	Relative catchability of stock `s` for fleet `f`. Defaults to 1 if missing. Matrix `[f, s]`
`delta`	Numeric, the duration of time in years corresponding to the observed catch, e.g., 0.25 is a quarterly time step.
`na`	Integer, number of age classes
`nr`	Integer, number of regions
`ns`	Integer, number of stocks
`nf`	Integer, number of fleets
`Fmax`	Numeric, the maximum Findex value
`nitF`	Integer, number of iterations for the Newton-Raphson routine
`trans`	Whether to perform the search in log or logit space

Details

The predicted catch for fleet `f` in region `r` is

$$C^{\text{pred}}_{f,r} = \sum_s \sum_a v_{a,f,s} q_{f,s} F_{f,r} \frac{1 - \exp(-Z_{a,r,s})}{Z_{a,r,s}} N_{a,r,s} w_{a,f,s}$$

The Newton-Raphson routine minimizes $f(x_{f,r}) = C^{\text{pred}}_{f,r} - C^{\text{obs}}_{f,r}$.

If `trans = "log"`, $F_{f,r} = \exp(x_{f,r})$.

If `trans = "logit"`, $F_{f,r} = F_{\max}/(1 + \exp(x_{f,r}))$.

The gradient with respect to $\vec{x}$ is

$$f'(x_{f,r}) = \sum_s \sum_a v_{a,f,s} q_{f,s} N_{a,r,s} w_{a,f,s} \left(\frac{\alpha\gamma}{\beta}\right)'$$

$$\left(\frac{\alpha\gamma}{\beta}\right)' = \frac{(\alpha\gamma' + \alpha'\gamma)\beta - \alpha\gamma\beta'}{\beta^2}$$

where

$$\begin{aligned}
&\alpha_{f,r} = F_{f,r} \\
&\beta_{a,r,s} = Z_{a,r,s} = M_{a,s} + \textstyle\sum_f v_{a,f,s} q_{f,s} F_{f,r} \\
&\gamma_{a,r,s} = 1 - \exp(-Z_{a,r,s}) \\
&\beta'_{a,f,r,s} = v_{a,f} q_{f,s} \alpha'_{f,r} \\
&\gamma'_{a,f,r,s} = \exp(-Z_{a,r,s}) \beta'_{a,f,r,s}
\end{aligned}$$

If `trans = "log"`, $\alpha'_{f,r} = \alpha_{f,r}$.

If `trans = "logit"`, $\alpha'_{f,r} = F_{\max} \exp(-x_{f,r})/(1 + \exp(-x_{f,r}))^2$.

This function solves for $\vec{x}$ by iterating until $f(\vec{x})$ approaches zero, where the vector arrow indexes over fleet and region. In iteration $i + 1$:

$$\vec{x}_{i+1} = \vec{x}_i - \frac{f(\vec{x}_i)}{f'(\vec{x}_i)}$$

.

Value

A list containing:

- `F_afrs` Fishing mortality array
- `F_ars` Fishing mortality array (summed across fleets)
- `Z_ars` Total mortality array
- `F_index` Index of fishing mortality. Matrix `[f, r]`
- `CB_frs` Catch (biomass) array
- `CN_afrs` Catch (abundance) array
- `VB_afrs` Vulnerable biomass at the beginning of the time step. Array
- `penalty` Penalty term returned by `posfun()` when `F_index` exceeds `Fmax`
- `fn` Difference between predicted and observed catch at the last iteration. Matrix `[f, r]`
- `gr` Gradient of `fn` with respect to `F_index` in either log or logit space at the last iteration. Vector by `[f, r]`

Author(s)

Q. Huynh

`calc_growth` *Calculate von Bertalanffy length-at-age*

Description

Returns an array of length-at-age with seasonal dimension. Useful for MSAdata inputs.

Usage

```
calc_growth(
  Linf_s,
  K_s,
  t0_s,
  ns = length(Linf_s),
  nm = 4,
  ny = 20,
  a = seq(1, 10) - 1
)
```

Arguments

`Linf_s`	Vector by stock `s` of asymptotic length
`K_s`	Vector by stock `s` of the growth coefficient
`t0_s`	Vector by `s` of the age at length zero.
`ns`	Integer, number of stocks
`nm`	Integer, number of seasons
`ny`	Integer, number of years
`a`	Integer vector of ages

Value

Array `[y, m, a, s]`

Examples

```
len_ymas <- calc_growth(c(30, 40), c(0.4, 0.2), c(-1, -1))

# Calculate stock weight at age
a_s <- rep(1e-6, 2)
b_s <- c(3, 3.1)

ns <- length(a_s)
swt_ymas <- sapply(1:ns, function(s) {
  a_s[s]*len_ymas[, , , s]^b_s[s]
}, simplify = "array")
```

`calc_index` *Calculate index at age*

Description

For indices of abundance, the function calculates the numbers vulnerable to the survey.

Usage

```
calc_index(
  N,
  Z,
  sel,
  na = dim(N)[1],
  nr = dim(N)[2],
  ns = dim(N)[3],
  ni = dim(sel)[2],
  samp = array(1, c(ni, nr, ns)),
  delta = rep(0, ni)
)
```

Arguments

`N`	Stock abundance at the beginning of the time step. Array `[a, r, s]`
`Z`	Instantaneous total mortality. Array `[a, r, s]`
`sel`	Index selectivity. Array `[a, i, s]`
`na`	Integer, number of age classes
`nr`	Integer, number of regions
`ns`	Integer, number of stocks
`ni`	Integer, number of indices
`samp`	Boolean indicates which regions and stocks are sampled by the index. Array `[i, r, s]`
`delta`	Fraction of time step when the index samples the population. Vector by `i`. Set to a negative number (-1) to sample the average population over the course of the time step, i.e. `N * (1 - exp(-Z))/Z`.

Details

The index is calculated as

$$I_{a,i,s} = v_{a,i,s} \sum_r N_{a,r,s} d_{a,r,s} \times \mathbb{1}(r \in R_i) \mathbb{1}(s \in S_i)$$

If the survey samples at a moment in time, then

$$d_{a,r,s} = \exp(-\delta_i Z_{a,r,s})$$

Otherwise, if the index samples the population over the duration of the time step, then

$$d_{a,r,s} = (1 - \exp(-Z_{a,r,s}))/Z_{a,r,s}$$

where R_i and S_i denote the regions and stocks, respectively, sampled by index i. For example, $R_2 = 1$ denotes that the second index of abundance only samples region 1. These are informed by array `samp` where `samp[i, r, s] = 1` indicates that stock `s` in region `r` is sampled by index `i`.

Value

Index at age. Array `[a, i, s]`

`calc_LAK` *Length-at-age key*

Description

Calculates the probability distribution of length-at-age using the normal probability density function

Usage

```
calc_LAK(len_a, sd_la, lbin, nl = length(lbin))
```

Arguments

`len_a`	Vector of length-at-age
`sd_la`	Vector of standard deviation in length-at-age
`lbin`	Vector of the lower boundary of the length bins
`nl`	Integer, number of length bins (default is `length(lbin)`)

Value

Matrix by age (rows) and length (columns)

`calc_nextN` *Project stock abundance to the next time step*

Description

This function applies survival of the current abundance, advances age classes, re-distributes the stock using the movement matrix.

Usage

```
calc_nextN(
  N,
  surv,
  na = dim(N)[1],
  nr = dim(N)[2],
  ns = dim(N)[3],
  advance_age = TRUE,
  mov = array(1/nr, c(na, nr, nr, ns)),
  plusgroup = TRUE
)
```

Arguments

`N`	Abundance at current time step. Array `[a, r, s]`
`surv`	Survival during the current time step. Array `[a, r, s]`
`na`	Integer, number of age classes
`nr`	Integer, number of regions
`ns`	Integer, number of stocks
`advance_age`	Logical, whether the animals advance to their next age class
`mov`	Movement array in the next time step. Array `[a, r, r, s]`. Rows denote region of origin and columns denote region of destination.
`plusgroup`	Logical, whether the last age class is an accumulator plus group.

Value

Abundance at the next time step. Array `[a, r, s]`

`calc_phi_project` *Equilibrium spawners per recruit by projection*

Description

Project a population forward in time using `calc_population()` with constant recruitment and seasonal dynamics (growth, movement-by-season) to obtain per recruit parameters. Note that the fishing mortality among fleets and stocks remain linked by matrix `q_fs`.

Usage

```
calc_phi_project(
  ny,
  nm,
  na,
  nf = 1,
  nr,
  ns = 1,
  F_mfr = array(0, c(nm, nf, nr)),
  sel_mafs = array(1, c(nm, na, nf, ns)),
  fwt_mafs = array(1, c(nm, na, nf, ns)),
  q_fs = matrix(1, nf, ns),
  M_as,
  mov_marrs,
  mat_as,
  fec_as,
  m_spawn = 1,
  m_advanceage = 1,
  delta_s = rep(0, ns),
  natal_rs = matrix(1, nr, ns),
  recdist_rs = matrix(1/nr, nr, ns)
)
```

Arguments

`ny`	Integer, number of years for the projection
`nm`	Integer, number of seasons
`na`	Integer, number of age classes
`nf`	Integer, number of fleets
`nr`	Integer, number of regions
`ns`	Integer, number of stocks
`F_mfr`	Equilibrium fishing mortality (per season). Matrix `[m, f, r]`
`sel_mafs`	Selectivity by season, age, fleet, stock. Array `[m, a, f, s]`
`fwt_mafs`	Fishery weight array by season, age, fleet, stock. Array `[m, a, r, r]`. Can be used calculate yield per recruit.

q_fs	Relative catchability of stock s for fleet f. Defaults to 1 if missing. Matrix [f, s]
M_as	Natural mortality. Matrix [a, s]
mov_marrs	Movement array [m, a, r, r, s]. If missing, uses a diagonal matrix (no movement among areas).
mat_as	Maturity at age. Matrix [a, s]
fec_as	Fecundity at age. Matrix [a, s]
m_spawn	Integer, season of spawning
m_advanceage	Integer, season at which to advance integer year age classes
delta_s	Numeric vector by s. Fraction of season that elapses when spawning occurs, e.g., midseason spawning when delta_s = 0.5.
natal_rs	Matrix [r, s]. The fraction of the mature stock s in region r that spawns at time of spawning. See example in Dstock.
recdist_rs	Matrix [r, s]. The fraction of the incoming recruitment of stock s that settles in region r.

Details

The initial population vector will be the survival at age evenly divided by the number of regions nr.

Value

A named list returned by calc_population().

See Also

calc_phi_simple()

---

calc_phi_simple *Simple spawners per recruit calculation*

Description

Calculate spawners per recruit by individual stock. Appropriate for a population model with a single spatial area and an annual time steps, i.e. single season.

Usage

```
calc_phi_simple(Z, fec_a, mat_a, delta = 0)

calc_NPR(Z, na = length(Z), plusgroup = TRUE)
```

Arguments

Z	Total mortality at age
fec_a	Fecundity at age. Vector
mat_a	Maturity at age. Vector
delta	Fraction of season that elapses when spawning occurs, e.g., midseason spawning when delta = 0.5.
na	Integer, number of age classes
plusgroup	Logical, whether the largest age class is a plusgroup accumulator age

Value

calc_phi_simple() returns a numeric for the spawning biomass per recruit.

calc_NPR() returns a numeric, numbers per recruit at age

See Also

calc_phi_project()

---

calc_POP *Predict the probability of CKMR kinship pairs*

Description

Calculate the probability of observing a parent-offspring pair (calc_POP) and half-sibling pair (calc_HSP) for closed-kin mark recapture (CKMR) for an age-structured model.

Usage

```
calc_POP(t, a, y, N, fec)

calc_HSP(yi, yj, N, fec, Z)
```

Arguments

t	Vector, capture year of parent i
a	Vector, age at capture of parent i
y	Vector, birth year of offspring j
N	Abundance of mature spawners. Matrix by [y, a]
fec	Fecundity schedule of mature spawners. Matrix by [y, a]
yi	Vector, birth year of sibling i. Must be older than sibling j.
yj	Vector, birth year of sibling j.
Z	Instantaneous total mortality rate. Matrix by [y, a]

Value

Numeric, vector of probabilities

Parent-offspring pairs

The parent-offspring probability is calculated from Bravington et al. 2016, eq 3.4:

$$p_{\text{POP}} = 2 \times \frac{f(y_j, y_j - (t_i - a_i))}{\sum_a f(y_j, a) N(y_j, a)}$$

where $y_j - (t_i - a_i)$ is the parental age in year y_j. Scalar 2 accounts for the fact that the parent could be either a mother or a father. `calc_POP` is vectorized with respect to `t`, `a`, and `y`.

Half-sibling pairs

The half-sibling probability is calculated from Bravington et al. 2016, eq 3.10, and expanded by Hillary et al. 2018, Supplement S2.8.1 for age-specific survival and fecundity of the parent:

$$p_{\text{HSP}} = 4 \times \sum_a \left(\frac{N(y_i, a) f(y_i, a)}{\sum_{a'} N(y_i, a') f(y_i, a')} \times \exp(- \sum_{t=0}^{y_j - y_i - 1} Z(y_i + t, a + t)) \times \frac{f(y_j, a + y_j - y_i)}{\sum_{a'} N(y_j, a') f(y_j, a')} \right)$$

- The first ratio is the probability that a fish at age a in year y_i is the parent of i.
- The exponential term is that fish's survival from year y_i to y_j.
- The second ratio is the probability that the parent of i, age $a + y_j - y_i$ in year y_j, is the parent of j.

The parent is not observed in the HSP, so we sum the probabilities over all potential ages in year y_i. `calc_HSP` is vectorized with respect to `yi` and `yj`.

Author(s)

Q. Huynh with contribution from Y. Tsukahara (Fisheries Research Institute, Japan)

References

Bravington, M.V. et al. 2016. Close-Kin Mark-Recapture. Stat. Sci. 31: 259-274. doi:10.1214/16STS552

Hillary, R.M. et al. 2018. Genetic relatedness reveals total population size of white sharks in eastern Australia and New Zealand. Sci. Rep. 8: 2661. doi:10.1038/s4159801820593w

See Also

`like_CKMR()`

`calc_population` *Multi-fleet, multi-area, multi-stock population dynamics model*

Description

Project age-structured populations forward in time. Also used by `[calc_phi_project()]` to calculate equilibrium abundance and biomass for which there is no analytic solution due to seasonal movement.

Usage

```
calc_population(
  ny = 10,
  nm = 4,
  na = 20,
  nf = 1,
  nr = 4,
  ns = 2,
  initN_ars = array(1, c(na, nr, ns)),
  mov_ymarrs,
  M_yas = array(0.3, c(ny, na, ns)),
  SRR_s = rep("BH", ns),
  sralpha_s = rep(1e+16, ns),
  srbeta_s = rep(1e+16, ns),
  mat_yas = array(1, c(ny, na, ns)),
  fec_yas = array(1, c(ny, na, ns)),
  Rdev_ys = matrix(1, ny, ns),
  m_spawn = 1,
  m_advanceage = 1,
  delta_s = rep(0, ns),
  natal_rs = matrix(1, nr, ns),
  recdist_rs = matrix(1/nr, nr, ns),
  fwt_ymafs = array(1, c(ny, nm, na, nf, ns)),
  q_fs = matrix(1, nf, ns),
  sel_ymafs = array(1, c(ny, nm, na, nf, ns)),
  condition = c("F", "catch"),
  F_ymfr = array(0, c(ny, nm, nf, nr)),
  Cobs_ymfr = array(1e-08, c(ny, nm, nf, nr)),
  Fmax = 2,
  nitF = 5L
)
```

Arguments

`ny`	Integer, number of years for the projection
`nm`	Integer, number of seasons

na	Integer, number of age classes
nf	Integer, number of fleets
nr	Integer, number of regions
ns	Integer, number of stocks
initN_ars	Abundance in the first year, first season. Array [a, r, s]
mov_ymarrs	Movement array [y, m, a, r, r, s]. If missing, uses a diagonal matrix (no movement among areas).
M_yas	Natural mortality (per year). Array [y, a, s]
SRR_s	Character vector by s for the stock recruit relationship. See calc_recruitment() for options
sralpha_s	Numeric vector by s for the stock recruit alpha parameter
srbeta_s	Numeric vector by s for the stock recruit beta parameter
mat_yas	Maturity ogive. Array [y, a, s]
fec_yas	Fecundity schedule (spawning output of mature individuals). Array [y, a, s]
Rdev_ys	Recruitment deviations. Matrix [y, s]
m_spawn	Integer, season of spawning
m_advanceage	Integer, season at which to advance integer year age classes
delta_s	Numeric vector by s. Fraction of season that elapses when spawning occurs, e.g., midseason spawning when delta_s = 0.5.
natal_rs	Matrix [r, s]. The fraction of the mature stock s in region r that spawns at time of spawning. See example in Dstock.
recdist_rs	Matrix [r, s]. The fraction of the incoming recruitment of stock s that settles in region r.
fwt_ymafs	Fishery weight at age. Array [y, m, a, f, s]
q_fs	Relative catchability of stock s for fleet f. Defaults to 1 if missing. Matrix [f, s]
sel_ymafs	Fishery selectivity. Array [y, m, a, f, s]
condition	Whether the fishing mortality is conditioned on the catch or specified F argument.
F_ymfr	Fishing mortality (per season). Array [y, m, f, r]. Only used if condition = "F".
Cobs_ymfr	Fishery catch (weight). Array [y, m, f, r]. Only used if condition = "catch" to solve for F (see calc_F()).
Fmax	Numeric, the maximum Findex value
nitF	Integer, number of iterations for the Newton-Raphson routine

Value

A named list containing:

- N_ymars Stock abundance
- F_ymars Fishing mortality (summed across fleets)
- F_ymfr Fishing mortality (by fleet and region)
- Z_ymars Total mortality
- F_ymafrs Fishing mortality (disaggregated by fleet)
- CN_ymafrs Catch at age (abundance)
- CB_ymfrs Fishery catch (weight)
- VB_ymfrs Vulnerable biomass available to the fishing fleets
- Nsp_yars Spawning abundance (in the spawning season)
- Npsp_yars Potentail spawners, mature animals in the spawning season that do not spawn if outside natal regions
- S_yrs Spawning output
- R_ys Recruitment
- penalty Numeric quadratic penalty if apical fishing mortality (by fleet) exceeds Fmax. See calc_F().

Examples

```
unfished_pop <- calc_population()
```

---

calc_recruitment *Calculate recruitment from stock-recruit function*

Description

Calculate recruitment from stock-recruit function

Usage

```
calc_recruitment(x, SRR = c("BH", "Ricker"), eq = FALSE, ...)
```

Arguments

x	Numeric, either the spawning output or the equilibrium spawners per recruit, from which the recruitment will be calculated. See argument eq.
SRR	Character to indicate the functional form of the stock recruit function
eq	Logical, indicates whether x is the spawning output (FALSE) or equilibrium spawners per recruit (TRUE)
...	Parameters of the SRR function. Provide one of two sets of variables: 1. h, R0 and phi0, or 2. a and b (alpha, beta values)

Value

Numeric of length x

Examples

```
calc_recruitment(10, SRR = "Ricker", a = 2, b = 0.5)
calc_recruitment(10, SRR = "Ricker", h = 0.9, R0 = 1, phi0 = 1)
```

---

check_data *Check dimensions and inputs in MSAdata object*

Description

Ensures that data inputs are of proper dimension. Whenever possible, default values are added to missing items.

Usage

```
check_data(MSAdata, silent = FALSE)
```

Arguments

MSAdata	S4 object containing data inputs. See MSAdata
silent	Logical, whether or not to report default values to the console

Value

An updated MSAdata object.

See Also

MSAdata

CondExpLt *If statements compatible with RTMB*

Description

Convenience functions that allow taping of gradients in RTMB with if expressions, following the corresponding CppAD functions.

Usage

```
CondExpLt(left, right, if_true, if_false)

CondExpLe(left, right, if_true, if_false)

CondExpGt(left, right, if_true, if_false)

CondExpGe(left, right, if_true, if_false)

CondExpEq(left, right, if_true, if_false)
```

Arguments

`left`	Numeric on left hand side of the evaluation
`right`	Numeric on right hand side of the evaluation
`if_true`	Numeric if expression is true
`if_false`	Numeric if expression is false

Details

Functions should be vectorized.

`CondExpLt` evaluates whether `left < right`

`CondExpLe` evaluates whether `left <= right`

`CondExpGt` evaluates whether `left > right`

`CondExpGe` evaluates whether `left >= right`

`CondExpEq` evaluates whether `left == right`

Value

Numeric

Examples

```
library(RTMB)
TapeConfig(comparison = "tape")
f <- function(x) CondExpLt(x, 3, 0, x^2)
g <- MakeTape(f, numeric(1))
x <- seq(0, 5)

# Does not work!
f2 <- function(x) if (x < 3) 0 else x^2
g2 <- MakeTape(f2, numeric(1))

# Compare the real answer (deriv) with various values returned by RTMB
data.frame(
  x = x,
  deriv = ifelse(x < 3, 0, 2 * x),
```

```
  deriv_f = sapply(x, g$jacobian),
  deriv_f2 = sapply(x, g2$jacobian)
)
```

---

conv_mov *Calculate movement matrix for all age classes*

Description

Movement matrices are calculated for all age classes from a base matrix and a gravity model formulation (Carruthers et al. 2016).

Usage

```
conv_mov(x, g, v, na = dim(x)[1], nr = dim(x)[2], aref = ceiling(0.5 * na))
```

Arguments

x	Base log-movement parameters. See details. Array [a, r, r]
g	Gravity model attractivity term. Tendency to move to region r. Matrix [a, r]
v	Gravity model viscosity term. Tendency to stay in same region. Vector by a
na	Integer, number of ages
nr	Integer, number of regions
aref	Integer, reference age class

Details

Rows index region of origin and columns denote region of destination.

In log space, the movement matrix m_a for age class a from region r to r' is the sum of base matrix x and gravity matrix G:

$$m_{a,r,r'} = x_{a,r,r'} + G_{a,r,r'}$$

To essentially exclude movement from r to r', set $x_{a,r,r'} = -1000$.

Gravity matrix G includes an attractivity term g and viscosity term v:

$$G_{a,r,r'} = \begin{cases} g'_{a,r'} + v_a & r = r' \\ g'_{a,r'} & \text{otherwise} \end{cases}$$

Vector g' are offset terms relative to the value for the reference age class:

$$g'_{a,r'} = \begin{cases} g_{a,r} & a = a_{ref} \\ g_{a,r} + g_{a=aref,r} & \text{otherwise} \end{cases}$$

The movement matrix in normal space is obtained by the softmax transformation:

$$M_{a,r,r'} = \frac{\exp(m_{a,r,r'})}{\sum_{r'} \exp(m_{a,r,r'})}$$

If x and v are zero, then the movement matrix simply distributes the total stock abundance into the various regions as specified in g'.

Value

Movement array `[a, r, r]`

References

Carruthers, T.R., et al. 2015. Modelling age-dependent movement: an application to red and gag groupers in the Gulf of Mexico. CJFAS 72: 1159-1176. doi:10.1139/cjfas20140471

`conv_selpar` *Selectivity at age and length*

Description

Calculate selectivity at age and length from a matrix of parameters.

- `conv_selpar()` converts parameters from log or logit space to real units.
- `calc_sel_len()` calculates selectivity at length.
- `calc_fsel_age()` calculates selectivity at age for fisheries, and coordinates dummy fleets.
- `calc_isel_age()` calculates selectivity at age for indices, and can map selectivity from fisheries or population parameters (e.g, mature or total biomass).

Usage

```
conv_selpar(x, type, maxage, maxL)

calc_sel_len(sel_par, lmid, type)

calc_fsel_age(
  sel_len,
  LAK,
  type,
  sel_par,
  sel_block = seq(1, length(type)),
  mat,
  a = seq(1, nrow(LAK)) - 1
)

calc_isel_age(
  sel_len,
  LAK,
  type,
  sel_par,
  fsel_age,
  maxage,
  mat,
  a = seq(1, nrow(LAK)) - 1
)
```

Arguments

x	Estimated parameters. Matrix [3, f]
type	Character string to indicate the functional form of selectivity. Options include: "logistic_length", "dome_length", "logistic_age", "dome_age", an integer (f) to map index selectivity to the corresponding fleet f (will be coerced to integer), "SB" to fix to maturity at age schedule, or "B" to fix to 1 for all ages.
maxage	Maximum value of the age of full selectivity
maxL	Maximum value of the length of full selectivity
sel_par	Matrix of parameters returned by conv_selpar()
lmid	Midpoints of length bins for calculating selectivity at length
sel_len	Selectivity at length matrix returned by calc_sel_len()
LAK	Length-at-age probability matrix. Matrix [a, length(lmid)]
sel_block	Integer vector. Length length(type). See details below.
mat	Maturity at age vector
a	Integer vector of ages corresponding to the rows of LAK (as well as mat)
fsel_age	Matrix returned by calc_fsel_age()

Value

conv_selpar() returns a matrix of the same dimensions as x.

calc_sel_len() returns a matrix [l, f], i.e., [length(lmid), length(type)].

calc_fsel_age() returns a matrix [a, f], i.e., [a, length(sel_block)]

calc_isel_age() returns a matrix [a, i], i.e., [a, length(type)]

Converting selectivity parameters (conv_selpar)

The first row of x corresponds to the length or age of full selectivity: $\mu_f = L_{max}/(1+\exp(-x_{1,f}))$

The second row of x corresponds to the width of the ascending limb for selectivity: $\sigma_f^{asc} = \exp(x_{2,f})$

The third row of x corresponds to the width of the descending limb for selectivity (if dome-shaped): $\sigma_f^{des} = \exp(x_{3,f})$

Length selectivity (calc_sel_len)

The selectivity at length is

$$s_\ell = \begin{cases} \exp(\alpha) & L_\ell < \mu_f \\ \exp(\beta) & L_\ell \geq \mu_f \end{cases}$$

where $\alpha = -0.5(L_\ell - \mu_f)^2/(\sigma_f^{asc})^2$ and $\beta = -0.5(L_\ell - \mu_f)^2/(\sigma_f^{des})^2$

Age selectivity (calc_fsel_age)

The equivalent selectivity at age is converted from the length values (`sel_len`) as

$$s_a = \sum_{\ell} s_\ell \times \text{Prob}(L_\ell | a)$$

If selectivity is explicitly in age units, then the values are directly calculated from parameters in `sel_par`.

Vector `sel_block` assigns the output selectivity from a different column of the input matrix and facilitates time-varying selectivity in time blocks. For example, `sel_block[1] <- 2` means that selectivity values in the first column of the output is based on the second column of the input matrices (`sel_len[, 2]` or `sel_par[, 2]`).

---

conv_Sigma *Calculate covariance matrix*

Description

Uses Cholesky factorization to generate a covariance matrix (or any symmetric positive definite matrix).

Usage

```
conv_Sigma(sigma, lower_diag)
```

Arguments

`sigma`	Numeric vector of marginal standard deviations (all greater than zeros)
`lower_diag`	Numeric vector to populate the lower triangle of the correlation matrix. All real numbers. Length `sum(1:(length(sigma) - 1))`

Value

Numeric

Examples

```
set.seed(23)
n <- 5
sigma <- runif(n, 0, 2)
lower_diag <- runif(sum(1:(n-1)), -10, 10)
Sigma <- conv_Sigma(sigma, lower_diag)
Sigma/t(Sigma) # Is symmetric matrix? All ones
cov2cor(Sigma)
```

DCKMR-class *DCKMR S4 object*

Description

Store lists of data frames for parent offspring pairs and half-sibling pairs

Slots inherited from DCKMR

`POP_s` A list by stock of data frames for parent-offspring pairs. Each row in the data frame corresponds to a "sampling unit" defined by the columns:

`a`	Capture year of parent
`t`	Age at capture of parent
`y`	Birth year of offspring
`n`	Number of pairwise comparisons
`m`	Number of POPs

`HSP_s` A list by stock of data frames for half-sibling pairs. Each row in the data frame corresponds to a "sampling unit" defined by the columns:

`yi`	Birth year of older sibling
`yj`	Birth year of younger sibling
`n`	Number of pairwise comparisons
`m`	Number of HSPs

`CKMR_like` Character, likelihood for the POP and HSP sampling units. See `type` argument in `like_CKMR()` for options.

Dfishery-class *Dfishery S4 object*

Description

S4 class that organizes the various data inputs for the MSA model. `MSAdata` simply inherits the slots from 6 component classes: `Dmodel`, `Dstock`, `Dfishery`, `Dsurvey` `DCKMR`, and `Dtag`, where the `D-` prefix denotes an object for data inputs (or model configuration).

Details

For convenience, most arrays and matrices have the associated dimensions in the variable name. For example, `Cobs_ymfr` represents observed catch with the dimension following the underscore, following this template:

`y`	Year
`m`	Season
`a`	Age
`r`	Region
`f`	Fishery
`i`	Index
`s`	Stock

Slots inherited from Dfishery

`nf` Integer, number of fleets

`Cobs_ymfr` Total fishery catch

`Csd_ymfr` Lognormal standard deviation of the fishery catch. Only used if `Dmodel@condition = "F"`. Default of 0.01.

`fwt_ymafs` Fishery weight at age. Set to 1 when fleet catch is in units of abundance. Set to stock weight at age by default (values at beginning of time step).

`CAAobs_ymafr` Fishery catch at age composition

`CALobs_ymlfr` Fishery catch at length composition

`fcomp_like` Character, likelihood for the fishery composition data. See `type` argument of `like_comp()` for options

`CAAN_ymfr` Sample size of the catch at age vector by season if using the multinomial or Dirichlet-multinomial likelihoods

`CALN_ymfr` Sample size of the catch at length vector by season if using the multinomial or Dirichlet-multinomial likelihoods

`CAAtheta_f` Catch at age dispersion parameter if using the Dirichlet-multinomial likelihood. Default set to 1.

`CALtheta_f` Catch at length dispersion parameter if using the Dirichlet-multinomial likelihood. Default set to 1.

`sel_block_yf` Index of dummy fleets to model time blocks of selectivity

`sel_f` Character vector of the functional form for selectivity. Choose between: `"logistic_length"`, `"dome_length"`, `"lo`

`Cinit_mfr` Equilibrium seasonal catch prior to the first year. One way to initialize the abundance at the start of the first year in the model. Default of zero.

`SC_ymafrs` Stock composition data.

`SC_aa` Boolean matrix that aggregates age classes for the stock composition data. See example.

`SC_ff` Boolean matrix that aggregates fleets for the stock composition data. See example.

`SC_like` Character, likelihood for the stock composition data. See `type` argument of `like_comp()` for options

SCN_ymafr Sample size of the stock composition vector if using the multinomial or Dirichlet-multinomial likelihoods

SCtheta_f Stock composition dispersion parameter if using the Dirichlet-multinomial likelihood. Default set to 1.

SCstdev_ymafrs Stock composition standard deviation if using the lognormal likelihood. Default set to 0.1.

See Also

MSAdata-class check_data() Dmodel-class Dstock-class Dfishery-class Dsurvey-class DCKMR-class Dtag-class

Examples

```
# Aggregate stock composition for ages 1-4 and 5-10 across all fleets
na <- 10
na_SC <- 2
SC_aa <- matrix(0, na_SC, na) # Assumes dim(SC_ymafrs)[3] = na_SC
SC_aa[1, 1:4] <- SC_aa[2, 5:10] <- 1

nf <- 3
nf_SC <- 1
SC_ff <- matrix(1, nf_SC, nf) # Assumes dim(SC_ymafrs)[4] <- nf_SC
```

Dlabel-class *Dlabel S4 object*

Description

Vectors for labeling plots.

Slots inherited from Dlabel

year Vector of years. Length Dmodel@ny

season Vector of season names. Length Dmodel@nm

age Vector of ages. Length Dmodel@na

region Vector of region names. Length Dmodel@nr

stock Vector of stock names. Length Dmodel@ns

fleet Vector of fleet names. Length Dfishery@nf

index Vector of index of abundance names. Length Dsurvey@ni

Dmodel-class *Dmodel S4 object*

Description

S4 class that organizes the various data inputs for the MSA model. MSAdata simply inherits the slots from 6 component classes: Dmodel, Dstock, Dfishery, Dsurvey DCKMR, and Dtag, where the D- prefix denotes an object for data inputs (or model configuration).

Details

For convenience, most arrays and matrices have the associated dimensions in the variable name. For example, Cobs_ymfr represents observed catch with the dimension following the underscore, following this template:

y	Year
m	Season
a	Age
r	Region
f	Fishery
i	Index
s	Stock

Slots inherited from Dmodel

ny Integer, number of years

nm Integer, number of seasons

na Integer, number of ages. The first age class is zero and the last age class (plus group is age na - 1).

nl Integer, number of length bins. Set to zero if lengths are not modeled.

nr Integer, number of spatial regions

ns Integer, number of stocks

lbin Vector of lower boundary of length bins. Length nl + 1

lmid Vector of midpoint of length bins. Length nl

Fmax Numeric, maximum allowable instantaneous fishing mortality rate (units of per season). Defaults to 3.

y_phi Integer, the year from which to obtain values of natural mortality and fecundity for the unfished stock-recruit replacement line (phi). Relevant if natural mortality or fecundity are time-varying. Defaults to 1.

scale_s Vector, length ns. Multiplicative scaling factor that informs relative stock size to aid parameter estimation. Larger values implies larger stocks. Default set to 1. See make_parameters().

nyinit Integer, number of years of spool-up to calculate equilibrium unfished and starting conditions for the population model to account for seasonal and spatial dynamics. The numerical spool-up is not needed when both nm = 1 and nr = 1, i.e., nyinit = 1. Otherwise, set to 2 * na by default.

`condition` Character, either to specify the model estimates fishing mortality as a parameter (`"F"`, default) or equal to the catch (`"catch"`).

`nitF` Integer, number of iterations to solve Baranov catch equation from observed catch if `condition = "catch"`. Defaults to 5.

`y_Fmult_f` Integer vector by fleet, the year in which to directly estimate F. Choose a year/season/region combination when the catch is average relative to the time series. Only used if `condition = "F"`.

`m_Fmult_f` Integer vector by fleet, the season in which to directly estimate F. Choose a year/season/region combination when the catch is average relative to the time series. Only used if `condition = "F"`.

`r_Fmult_f` Integer vector by fleet, the region in which to directly estimate F. Choose a year/season/region combination when the catch is average relative to the time series. Only used if `condition = "F"`.

`pbc_rdev_ys` Numeric matrix, for the fraction of lognormal bias correction (`-0.5 * sd_r^2`) applied to the recruitment estimates in the model. Typically between 0-1, with default of 1.

`pbc_initrdev_as` Numeric matrix, for the fraction of lognormal bias correction (`-0.5 * sd_r^2`) applied to the initial abundance vector in the model. Typically between 0-1, with default of 1.

`prior` Character vector to be evaluated in the model to return the log prior for a parameter. See example in documentation for prior.

`nyret` Integer, number of recent years of data to remove from the likelihood for retrospective analysis (positive numbers). Default is zero.

See Also

MSAdata-class `check_data()` Dmodel-class Dstock-class Dfishery-class Dsurvey-class DCKMR-class Dtag-class

---

`Dstock-class` *Dstock S4 object*

Description

S4 class that organizes the various data inputs for the MSA model. `MSAdata` simply inherits the slots from 6 component classes: `Dmodel`, `Dstock`, `Dfishery`, `Dsurvey` `DCKMR`, and `Dtag`, where the `D-` prefix denotes an object for data inputs (or model configuration).

Details

For convenience, most arrays and matrices have the associated dimensions in the variable name. For example, `Cobs_ymfr` represents observed catch with the dimension following the underscore, following this template:

`y`	Year
`m`	Season

a Age
r Region
f Fishery
i Index
s Stock

Slots inherited from Dstock

`m_spawn` Integer, season of spawning. Defaults to 1. The progeny will enter the age structure in the same season.

`m_advanceage` Integer, season in which to advance age classes (corresponding to calendar year) to the next age class. Defaults to 1.

`len_ymas` Length-at-age. Only needed if `Dmodel@nl > 0` to fit length composition (you may want to calculate the growth at the middle of the time step). `calc_growth()` may be a helpful function.

`sdlen_ymas` Standard deviation in length-at-age

`LAK_ymals` Length-at-age probability array. If empty, values will be calculated by `check_data()` with `calc_LAK()`.

`mat_yas` Proportion mature by age class. Ignored if maturity ogive is estimated, e.g., when fitting to close-kin genetic data.

`swt_ymas` Stock weight-at-age (at beginning of time step). See `calc_growth()` example.

`fec_yas` Fecundity, i.e., spawning output, of mature animals. Default uses stock weight at age.

`M_yas` Natural mortality. Instantaneous units per year. Ignored if M is estimated.

`SRR_s` Character vector of stock-recruit relationship by stock. See `SRR` argument in `calc_recruitment()` for options.

`delta_s` Fraction of season that elapses when spawning occurs, e.g., midseason spawning occurs when `delta_s = 0.5`. Default is zero.

`presence_rs` Logical matrix indicating presence/absence of stock `s` in region `r`. Used to constrain movement matrix. Default is TRUE for all stocks and regions.

`natal_rs` The fraction of the mature stock `s` in region `r` that spawns at time of spawning. See example. Default is 1 for all stocks and regions.

See Also

MSAdata-class `check_data()` Dmodel-class Dstock-class Dfishery-class Dsurvey-class DCKMR-class Dtag-class

Examples

```
# Set natal_rs matrix so that the spawning output of stock 1 is
# calculated from mature animals present in regions 1, 2.
# Similarly for stock 2, spawning output from areas 2 and 3.
nr <- 4
ns <- 2
natal_rs <- matrix(0, nr, ns)
natal_rs[1:2, 1] <- natal_rs[2:3, 2] <- 1
```

`Dsurvey-class` *Dsurvey S4 object*

Description

S4 class that organizes the various data inputs for the MSA model. `MSAdata` simply inherits the slots from 6 component classes: `Dmodel`, `Dstock`, `Dfishery`, `Dsurvey` `DCKMR`, and `Dtag`, where the `D-` prefix denotes an object for data inputs (or model configuration).

Details

For convenience, most arrays and matrices have the associated dimensions in the variable name. For example, `Cobs_ymfr` represents observed catch with the dimension following the underscore, following this template:

`y`	Year
`m`	Season
`a`	Age
`r`	Region
`f`	Fishery
`i`	Index
`s`	Stock

Slots inherited from Dsurvey

`ni` Integer, number of indices of abundance. Zero is possible.

`Iobs_ymi` Observed indices

`Isd_ymi` Lognormal standard deviation of the observed indices

`unit_i` Character vector, units of the index. Set to `"B"` to use stock weight at age (default) or `"N"` for abundance (numbers).

`IAAobs_ymai` Array, survey age composition

`IALobs_ymli` Array, survey length composition

`icomp_like` Character, likelihood for the composition data. See `like_comp()` for options

`IAAN_ymi` Array, sample size of the index age composition by season if using the multinomial or Dirichlet-multinomial likelihoods

`IALN_ymi` Array, sample size of the index length composition by season if using the multinomial or Dirichlet-multinomial likelihoods

`IAAtheta_i` Numeric vector, survey age composition dispersion parameter if using the Dirichlet-multinomial likelihood

`IALtheta_i` Numeric vector, index length composition dispersion parameter if using the Dirichlet-multinomial likelihood

`samp_irs` Boolean array that specifies the regions and stocks sampled by the index. `samp[i, r, s]` indicates whether index `i` operates in region `r` and catches stock `s`.

`sel_i` Character vector, functional forms for selectivity. See `"type"` argument in `conv_selpar()` for options.

`delta_i` Numeric vector, The elapsed fraction of time in the seasonal time step (between 0 - 1) when the index samples the population. Set to a negative number (-1) to sample over the duration of the timestep, i.e., `(1 - exp(-Z))/Z`.

See Also

MSAdata-class `check_data()` Dmodel-class Dstock-class Dfishery-class Dsurvey-class DCKMR-class Dtag-class

---

`Dtag-class` *Dtag S4 object*

Description

S4 class that organizes the various data inputs for the MSA model. `MSAdata` simply inherits the slots from 6 component classes: `Dmodel`, `Dstock`, `Dfishery`, `Dsurvey` `DCKMR`, and `Dtag`, where the `D-` prefix denotes an object for data inputs (or model configuration).

Details

For convenience, most arrays and matrices have the associated dimensions in the variable name. For example, `Cobs_ymfr` represents observed catch with the dimension following the underscore, following this template:

`y`	Year
`m`	Season
`a`	Age
`r`	Region
`f`	Fishery
`i`	Index
`s`	Stock

Slots inherited from Dtag

`tag_ymarrs` Array. Number of tags that move between regions. Informs movement matrices of stocks between time steps.

`tag_ymars` Array. Number of tags distributed among regions. Informs stock distribution (within time step).

`tag_yy` Boolean matrix that aggregates years for the tag data. Only used for the tag movement array `tag_ymarrs`.

`tag_aa` Boolean matrix that aggregates ages for the tag data.

`tag_like` Character. Likelihood for the tagging data, either the vector of proportions by region of origin for `tag_ymarrs`, or by region of stock distribution for `tag_ymars`. See `type` argument of `like_comp()` for options

`tagN_ymars` Array. Sample size of the tag movement vectors if using the multinomial or Dirichlet-multinomial likelihoods.

`tagN_ymas` Array. Sample size of the tag distribution vectors if using the multinomial or Dirichlet-multinomial likelihoods.

`tagtheta_s` Array. Tag dispersion parameter (by stock) if using the Dirichlet-multinomial likelihoods. Default set to 1.

`tagstdev_s` Array. Tag standard deviation (by stock) if using the lognormal likelihood. Default set to 0.1.

See Also

MSAdata-class `check_data()` Dmodel-class Dstock-class Dfishery-class Dsurvey-class DCKMR-class Dtag-class

---

`fit_MSA` *Fit MSA model*

Description

Wrapper function that calls RTMB to create the model and perform the numerical optimization

Usage

```
fit_MSA(
  MSAdata,
  parameters,
  map = list(),
  random = NULL,
  run_model = TRUE,
  do_sd = TRUE,
  report = TRUE,
  silent = FALSE,
  control = list(iter.max = 2e+05, eval.max = 4e+05),
  ...
)
```

Arguments

`MSAdata`	Data object. Class MSAdata, validated by `check_data()`
`parameters`	List of parameters, e.g., returned by `make_parameters()` and validated by `check_parameters()`.
`map`	List of parameters indicated whether they are fixed and how they are shared, e.g., returned by `make_parameters()`. See `RTMB::MakeADFun()`.
`random`	Character vector indicating the parameters that are random effects, e.g., returned by `make_parameters()`.
`run_model`	Logical, whether to fit the model through `stats::nlminb()`.

do_sd	Logical, whether to calculate the standard errors with RTMB::sdreport().
report	Logical, whether to return the report list with obj$report(obj$env$last.par.best).
silent	Logical, whether to report progress to console. **Not passed to TMB::MakeADFun().** Recommend to set to TRUE to speed up run time, e.g., when running simulations, multiple fits, etc.
control	Passed to stats::nlminb()
...	Other arguments to RTMB::MakeADFun().

Value

A MSAassess object.

See Also

report() retrospective()

---

get_MSAdata *Retrieve data object used to fit model*

Description

A convenient function to retrieve the data object used to fit the model. The object is embedded in an environment within the RTMB object.

Usage

```
get_MSAdata(MSAassess)
```

Arguments

MSAassess	MSAassess object returned by fit_MSA()

Value

MSAdata object

`get_sdreport` *Calculate standard errors*

Description

A wrapper function to return standard errors. Various numerical techniques are employed to obtain a positive-definite covariance matrix.

Usage

```
get_sdreport(obj, getReportCovariance = FALSE, silent = FALSE, ...)
```

Arguments

`obj`	The list returned by `RTMB::MakeADFun()`
`getReportCovariance`	Logical, passed to `RTMB::sdreport()`
`silent`	Logical, whether to report progress to console. See details.
`...`	Other arguments to `RTMB::sdreport()` besides `par.fixed, hessian.fixed, getReportCovariance`

Details

In marginal cases where the determinant of the Hessian matrix is less than 0.1, several steps are utilized to obtain a positive-definite covariance matrix, in the following order:

1. Invert hessian returned by `h <- obj@he(obj$env.last.par.best)` (skipped in models with random effects)
2. Invert hessian returned by `h <- stats::optimHess(obj$env.last.par.best, obj$fn, obj$gr)`
3. Invert hessian returned by `h <- numDeriv::jacobian(obj$gr, obj$env.last.par.best)`
4. Calculate covariance matrix from `chol2inv(chol(h))`

Value

Object returned by `RTMB::sdreport()`. A correlation matrix is generated and stored in: `get_sdreport(obj)$env$corr.fi`

like_CKMR *Likelihood for CKMR*

Description

Returns the log-likelihood for a set of pairwise comparisons. For a parent-offspring pair, a comparison is defined by the capture year of parent, capture age of parent, and birth year of offspring. For a half-sibling pair, a comparison is defined by the cohort year of each sibling. Binomial and Poisson distributions are supported (Conn et al. 2020).

Usage

```
like_CKMR(n, m, p, type = c("binomial", "poisson"))
```

Arguments

n	The number of pairwise comparisons
m	The number of kinship matches
p	The probability of kinship match
type	The statistical distribution for the likelihood calculation

Value

Numeric representing the log-likelihood.

References

Conn, P.B. et al. 2020. Robustness of close-kin mark-recapture estimators to dispersal limitation and spatially varying sampling probabilities. Ecol. Evol. 10: 5558-5569. doi:10.1002/ece3.6296

See Also

calc_POP() calc_HSP()

like_comp *Likelihood for composition vectors*

Description

Returns the log-likelihood for composition data, e.g., length, age, or stock composition, with various statistical distributions supported.

Usage

```
like_comp(
  obs,
  pred,
  type = c("multinomial", "dirmult1", "dirmult2", "lognormal"),
  N = sum(obs),
  theta,
  stdev
)
```

Arguments

obs	A vector of observed values. Internally converted to proportions.
pred	A vector of predicted values. Same length as obs. Internally converted to proportions.
type	Character for the desired distribution
N	Numeric, the sample size corresponding to obs for multinomial or Dirichlet multinomial distributions.
theta	Numeric, the linear (type = "dirmult1") or saturating (type = "dirmult2") Dirichlet-multinomial parameter, respectively. See Thorson et al. (2017)
stdev	Numeric or vectorized for obs, the likelihood standard deviation for lognormal distribution.

Details

Observed and predicted vectors are internally converted to proportions.

For type = "lognormal", zero observations are removed from the likelihood calculation.

Value

Numeric representing the log-likelihood.

References

Thorson et al. 2017. Model-based estimates of effective sample size in stock assessment models using the Dirichlet-multinomial distribution. Fish. Res. 192:84-93. doi:10.1016/j.fishres.2016.06.005

Examples

```
M <- 0.1
age <- seq(1:10)
pred <- exp(-M * age)
obs <- pred * rlnorm(10, sd = 0.05)
like_comp(obs, pred, N = 10, type = "multinomial")
like_comp(obs, pred, N = 100, type = "multinomial")
like_comp(obs, pred, N = 10, type = "dirmult1", theta = 1)
like_comp(obs, pred, N = 10, type = "dirmult1", theta = 20)
```

make_parameters *Make list of parameters for RTMB*

Description

Sets up the list of parameters, map of parameters (see map argument in TMB::MakeADFun()), and identifies some random effects parameters based on the input data and some user choices on model configuration.

These functions provide a template for the parameter and map setup that can be adjusted for alternative configurations. check_parameters() checks whether custom made parameter lists are of the correct dimension.

Usage

```
make_parameters(
  MSAdata,
  start = list(),
  map = list(),
  est_mov = c("none", "dist_random", "gravity_fixed"),
  silent = FALSE,
  ...
)

make_map(
  p,
  MSAdata,
  map = list(),
  est_M = FALSE,
  est_h = FALSE,
  est_mat = FALSE,
  est_sdr = FALSE,
  est_mov = c("none", "dist_random", "gravity_fixed"),
  est_qfs = FALSE,
  silent = FALSE
)

check_parameters(p = list(), map, MSAdata, silent = FALSE)
```

Arguments

MSAdata	S4 data object
start	An optional list of parameters. Named list of parameters with the associated dimensions and transformations below. Overrides default values created by make_parameters().
map	List of mapped parameters. Used by check_parameters() only to count parameters.

est_mov	Character describing structure of stock movement parameters. Only used if map arguments for the movement parameters is NULL. See details below.
silent	Logical, whether make_map() reports messages to the console
...	Various arguments for make_map() (could be important!)
p	List of parameters, e.g., returned by make_parameters()
est_M	Logical, estimate natural mortality? Only used if map$log_M_s is NULL
est_h	Logical, estimate steepness? Only used if map$t_h_s is NULL
est_mat	Logical, estimate maturity? Only used if map$mat_ps is NULL
est_sdr	Logical, estimate standard deviation of recruitment deviates? Only used if map$log_sdr_s is NULL
est_qfs	Logical, estimate relative catchability of stocks by each fleet? Fix log_q_fs for the first stock if TRUE. Only used if map$log_q_fs is NULL

Value

make_parameters() returns a list of parameters ("p") concatenated with the output of make_map().

make_map() returns a named list containing parameter mappings ("map") and a character vector of random effects ("random").

check_parameters() invisibly returns the parameter list if no problems are encountered.

Parameters

Generally parameter names will have up to three components, separated by underscores. For example, log_M_s represents the natural logarithm of natural mortality, and is a vector by stock s.

The first component describes the transformation from the estimated parameter space to the normal parameter space, frequently, log or logit. Prefix t indicates some other custom transformation that is described below.

Second is the parameter name, e.g., M for natural mortality, rdev for recruitment deviates, etc.

Third is the dimension of the parameter variable and the indexing for the vectors, matrices, and arrays, e.g., y for year, s for stock. See MSAdata. Here, an additional index p represents some other number of parameters that is described below.

t_R0_s Vector by s. Unfished recruitment, i.e., intersection of unfished replacement line and average stock recruit function, is represented as: R0_s <- exp(t_R0_s) * MSAdata@Dmodel@scale_s. By default, t_R0_s = 3

t_h_s Vector by s. Steepness of the stock-recruit function. Logit space for Beverton-Holt and log space for Ricker functions. Default steepness value of 0.8

mat_ps Matrix [2, s]. Maturity parameters (can be estimated or specified in data object). Logistic functional form. The parameter in the first row is the age of 50 percent maturity in logit space: a50_s <- plogis(mat_ps[1,] * na). In the second row is the age of 95 percent maturity as a logarithmic offset: a95_s <- a50_s + exp(mat_ps[2,]). Default a50_s <- 0.5 * na and a95_s <- a50_s + 1

log_M_s Vector by s. Natural logarithm of natural mortality (can be estimated or specified in data object). Default parameter value for all stocks: M <- -log(0.05)/MSAdata@Dmodel@na

`log_rdev_ys` Matrix `[y, s]`. Log recruitment deviations. By default, all start values are at zero.

`log_sdr_s` Vector by `s`. log-Standard deviation of the log recruitment deviations. Default SD = 0.4

`log_q_fs` Matrix `[f, s]`. The natural logarithm of `q_fs`, the relative fishing efficiency of `f` for stock `s`. Equal values imply equal catchability of all stocks. See equations in `calc_F()`. Default sets all values to zero.

`log_Fdev_ymfr` Array `[y, m, f, r]`. Fishing mortality parameters. For each fleet, the log of F is estimated directly for the reference year, season, region. For other strata, F is an offset from this value:

$$F_{y,m,f,r} = \begin{cases} \exp(x_f^{\text{Fmult}}) & y = y_{\text{ref}}, m = m_{\text{ref}}, r = r_{\text{ref}} \\ \exp(x_f^{\text{Fmult}} + x_{y,m,r}^{\text{Fdev}}) & \text{otherwise} \end{cases}$$

`sel_pf` Matrix `[3, f]`. Fishery selectivity parameters in logit or log space. See equations `conv_selpar()`, where `sel_pf` is the `x` matrix.

`sel_pi` Matrix `[3, i]`. Index selectivity parameters in logit or log space. See equations `conv_selpar()`, where `sel_pi` is the `x` matrix.

`mov_x_marrs` Array `[m, a, r, r, s]`. Base movement matrix. Set to -1000 to effectively exclude movements from region pairs. See equations in `conv_mov()`

`mov_g_ymars` Array `[y, m, a, r, s]`. Attractivity term in gravity model for movement. If `x` and `v` are zero, this matrix specifies the distribution of total stock abundance into the various regions. See equations in `conv_mov()`

`mov_v_ymas` Array `[y, m, a, s]`. Viscosity term in gravity model for movement. See equations in `conv_mov()`

`log_sdg_rs` Array `[r, s]`. Marginal log standard deviation in the stock distribution (`mov_g_ymars`) among regions for stock `s`. Only used when `est_mov = "dist_random"`. Default SD of 0.1.

`t_corg_ps` Array `[sum(1:(nr - 1)), s]`. Lower triangle of the correlation matrix for `mov_g_ymars`, to be obtained with the Cholesky factorization. Only used when `est_mov = dist_random`. Default values of zero.

`log_initF_mfr` Array `[m, f, r]`. Initial F corresponding to the equilibrium catch.

`log_initrdev_as` Array `[na - 1, s]`. Recruitment deviations for the initial abundance-at-age vector.

Start list

Users can provide `R0_s` and `h_s` in the start list. `make_parameters()` will make the appropriate transformation for the starting values of `t_R0_s` and `t_h_s`, respectively.

Movement setup for `make_map()`

If a single region model or `est_mov = "none"`: no movement parameters are estimated.

If `est_mov = "dist_random"`: fix all values for `mov_x_marrs` and `mov_v_ymas`. Fix `mov_g_ymars` for the first region for each year, season, age, and stock. `mov_g_ymars` are random effects.

If `est_mov = "gravity_fixed"`: fix all values for `mov_x_marrs`. Fix `mov_g_ymars` for the first region for each year, season, age, and stock. Estimate all `mov_v_ymas`. Both `mov_g_ymars` and `mov_v_ymas` are fixed effects.

By default p$mov_x_marrs is zero. Set to -1000 for areas for which there is no abundance of a particular stock.

See Also

MSAdata

MSAassess-class *MSAassess S4 object*

Description

S4 object that returns output from MSA model

Slots

obj RTMB object returned by RTMB::MakeADFun()

opt List returned by stats::nlminb()

SD List returned by RTMB::sdreport()

report List of model output at the parameter estimates, returned by obj$report(obj$env$last.par.best)

Misc List, miscellaneous items

MSAdata-class *MSAdata S4 object*

Description

S4 class that organizes the various data inputs for the MSA model. MSAdata simply inherits the slots from 6 component classes: Dmodel, Dstock, Dfishery, Dsurvey DCKMR, and Dtag, where the D- prefix denotes an object for data inputs (or model configuration).

Details

For convenience, most arrays and matrices have the associated dimensions in the variable name. For example, Cobs_ymfr represents observed catch with the dimension following the underscore, following this template:

y	Year
m	Season
a	Age
r	Region
f	Fishery
i	Index
s	Stock

Slots

`Dmodel` Class Dmodel containing parameters for model structure (number of years, ages, etc.)

`Dstock` Class Dstock containing stock parameters (growth, natural mortality, etc.)

`Dfishery` Class Dfishery containing fishery data (catch, size and stock composition, etc.)

`Dsurvey` Class Dsurvey containing survey data (indices of abundance)

`DCKMR` Class DCKMR containing genetic close-kin data

`Dtag` Class Dtag containing tagging data

`Dlabel` Class Dlabel containing names for various dimensions. Used for plotting.

`Misc` List for miscellaneous inputs as needed

Slots inherited from Dmodel

`ny` Integer, number of years

`nm` Integer, number of seasons

`na` Integer, number of ages. The first age class is zero and the last age class (plus group is age `na - 1`).

`nl` Integer, number of length bins. Set to zero if lengths are not modeled.

`nr` Integer, number of spatial regions

`ns` Integer, number of stocks

`lbin` Vector of lower boundary of length bins. Length `nl + 1`

`lmid` Vector of midpoint of length bins. Length `nl`

`Fmax` Numeric, maximum allowable instantaneous fishing mortality rate (units of per season). Defaults to 3.

`y_phi` Integer, the year from which to obtain values of natural mortality and fecundity for the unfished stock-recruit replacement line (`phi`). Relevant if natural mortality or fecundity are time-varying. Defaults to 1.

`scale_s` Vector, length `ns`. Multiplicative scaling factor that informs relative stock size to aid parameter estimation. Larger values implies larger stocks. Default set to 1. See `make_parameters()`.

`nyinit` Integer, number of years of spool-up to calculate equilibrium unfished and starting conditions for the population model to account for seasonal and spatial dynamics. The numerical spool-up is not needed when both `nm = 1` and `nr = 1`, i.e., `nyinit = 1`. Otherwise, set to `2 * na` by default.

`condition` Character, either to specify the model estimates fishing mortality as a parameter (`"F"`, default) or equal to the catch (`"catch"`).

`nitF` Integer, number of iterations to solve Baranov catch equation from observed catch if `condition = "catch"`. Defaults to 5.

`y_Fmult_f` Integer vector by fleet, the year in which to directly estimate F. Choose a year/season/region combination when the catch is average relative to the time series. Only used if `condition = "F"`.

`m_Fmult_f` Integer vector by fleet, the season in which to directly estimate F. Choose a year/season/region combination when the catch is average relative to the time series. Only used if `condition = "F"`.

r_Fmult_f Integer vector by fleet, the region in which to directly estimate F. Choose a year/season/region combination when the catch is average relative to the time series. Only used if condition = "F".

pbc_rdev_ys Numeric matrix, for the fraction of lognormal bias correction (-0.5 * sd_r^2) applied to the recruitment estimates in the model. Typically between 0-1, with default of 1.

pbc_initrdev_as Numeric matrix, for the fraction of lognormal bias correction (-0.5 * sd_r^2) applied to the initial abundance vector in the model. Typically between 0-1, with default of 1.

prior Character vector to be evaluated in the model to return the log prior for a parameter. See example in documentation for prior.

nyret Integer, number of recent years of data to remove from the likelihood for retrospective analysis (positive numbers). Default is zero.

Slots inherited from Dstock

m_spawn Integer, season of spawning. Defaults to 1. The progeny will enter the age structure in the same season.

m_advanceage Integer, season in which to advance age classes (corresponding to calendar year) to the next age class. Defaults to 1.

len_ymas Length-at-age. Only needed if Dmodel@nl > 0 to fit length composition (you may want to calculate the growth at the middle of the time step). calc_growth() may be a helpful function.

sdlen_ymas Standard deviation in length-at-age

LAK_ymals Length-at-age probability array. If empty, values will be calculated by check_data() with calc_LAK().

mat_yas Proportion mature by age class. Ignored if maturity ogive is estimated, e.g., when fitting to close-kin genetic data.

swt_ymas Stock weight-at-age (at beginning of time step). See calc_growth() example.

fec_yas Fecundity, i.e., spawning output, of mature animals. Default uses stock weight at age.

M_yas Natural mortality. Instantaneous units per year. Ignored if M is estimated.

SRR_s Character vector of stock-recruit relationship by stock. See SRR argument in calc_recruitment() for options.

delta_s Fraction of season that elapses when spawning occurs, e.g., midseason spawning occurs when delta_s = 0.5. Default is zero.

presence_rs Logical matrix indicating presence/absence of stock s in region r. Used to constrain movement matrix. Default is TRUE for all stocks and regions.

natal_rs The fraction of the mature stock s in region r that spawns at time of spawning. See example. Default is 1 for all stocks and regions.

Slots inherited from Dfishery

nf Integer, number of fleets

Cobs_ymfr Total fishery catch

Csd_ymfr Lognormal standard deviation of the fishery catch. Only used if Dmodel@condition = "F". Default of 0.01.

`fwt_ymafs` Fishery weight at age. Set to 1 when fleet catch is in units of abundance. Set to stock weight at age by default (values at beginning of time step).

`CAAobs_ymafr` Fishery catch at age composition

`CALobs_ymlfr` Fishery catch at length composition

`fcomp_like` Character, likelihood for the fishery composition data. See `type` argument of `like_comp()` for options

`CAAN_ymfr` Sample size of the catch at age vector by season if using the multinomial or Dirichlet-multinomial likelihoods

`CALN_ymfr` Sample size of the catch at length vector by season if using the multinomial or Dirichlet-multinomial likelihoods

`CAAtheta_f` Catch at age dispersion parameter if using the Dirichlet-multinomial likelihood. Default set to 1.

`CALtheta_f` Catch at length dispersion parameter if using the Dirichlet-multinomial likelihood. Default set to 1.

`sel_block_yf` Index of dummy fleets to model time blocks of selectivity

`sel_f` Character vector of the functional form for selectivity. Choose between: `"logistic_length"`, `"dome_length"`, `"lo`

`Cinit_mfr` Equilibrium seasonal catch prior to the first year. One way to initialize the abundance at the start of the first year in the model. Default of zero.

`SC_ymafrs` Stock composition data.

`SC_aa` Boolean matrix that aggregates age classes for the stock composition data. See example.

`SC_ff` Boolean matrix that aggregates fleets for the stock composition data. See example.

`SC_like` Character, likelihood for the stock composition data. See `type` argument of `like_comp()` for options

`SCN_ymafr` Sample size of the stock composition vector if using the multinomial or Dirichlet-multinomial likelihoods

`SCtheta_f` Stock composition dispersion parameter if using the Dirichlet-multinomial likelihood. Default set to 1.

`SCstdev_ymafrs` Stock composition standard deviation if using the lognormal likelihood. Default set to 0.1.

Slots inherited from Dsurvey

`ni` Integer, number of indices of abundance. Zero is possible.

`Iobs_ymi` Observed indices

`Isd_ymi` Lognormal standard deviation of the observed indices

`unit_i` Character vector, units of the index. Set to `"B"` to use stock weight at age (default) or `"N"` for abundance (numbers).

`IAAobs_ymai` Array, survey age composition

`IALobs_ymli` Array, survey length composition

`icomp_like` Character, likelihood for the composition data. See `like_comp()` for options

`IAAN_ymi` Array, sample size of the index age composition by season if using the multinomial or Dirichlet-multinomial likelihoods

IALN_ymi Array, sample size of the index length composition by season if using the multinomial or Dirichlet-multinomial likelihoods

IAAtheta_i Numeric vector, survey age composition dispersion parameter if using the Dirichlet-multinomial likelihood

IALtheta_i Numeric vector, index length composition dispersion parameter if using the Dirichlet-multinomial likelihood

samp_irs Boolean array that specifies the regions and stocks sampled by the index. samp[i, r, s] indicates whether index i operates in region r and catches stock s.

sel_i Character vector, functional forms for selectivity. See "type" argument in conv_selpar() for options.

delta_i Numeric vector, The elapsed fraction of time in the seasonal time step (between 0 - 1) when the index samples the population. Set to a negative number (-1) to sample over the duration of the timestep, i.e., (1 - exp(-Z))/Z.

Slots inherited from DCKMR

POP_s A list by stock of data frames for parent-offspring pairs. Each row in the data frame corresponds to a "sampling unit" defined by the columns:

a	Capture year of parent
t	Age at capture of parent
y	Birth year of offspring
n	Number of pairwise comparisons
m	Number of POPs

HSP_s A list by stock of data frames for half-sibling pairs. Each row in the data frame corresponds to a "sampling unit" defined by the columns:

yi	Birth year of older sibling
yj	Birth year of younger sibling
n	Number of pairwise comparisons
m	Number of HSPs

CKMR_like Character, likelihood for the POP and HSP sampling units. See type argument in like_CKMR() for options.

Slots inherited from Dtag

tag_ymarrs Array. Number of tags that move between regions. Informs movement matrices of stocks between time steps.

tag_ymars Array. Number of tags distributed among regions. Informs stock distribution (within time step).

tag_yy Boolean matrix that aggregates years for the tag data. Only used for the tag movement array tag_ymarrs.

tag_aa Boolean matrix that aggregates ages for the tag data.

`tag_like` Character. Likelihood for the tagging data, either the vector of proportions by region of origin for `tag_ymarrs`, or by region of stock distribution for `tag_ymars`. See `type` argument of `like_comp()` for options

`tagN_ymars` Array. Sample size of the tag movement vectors if using the multinomial or Dirichlet-multinomial likelihoods.

`tagN_ymas` Array. Sample size of the tag distribution vectors if using the multinomial or Dirichlet-multinomial likelihoods.

`tagtheta_s` Array. Tag dispersion parameter (by stock) if using the Dirichlet-multinomial likelihoods. Default set to 1.

`tagstdev_s` Array. Tag standard deviation (by stock) if using the lognormal likelihood. Default set to 0.1.

Slots inherited from Dlabel

`year` Vector of years. Length `Dmodel@ny`

`season` Vector of season names. Length `Dmodel@nm`

`age` Vector of ages. Length `Dmodel@na`

`region` Vector of region names. Length `Dmodel@nr`

`stock` Vector of stock names. Length `Dmodel@ns`

`fleet` Vector of fleet names. Length `Dfishery@nf`

`index` Vector of index of abundance names. Length `Dsurvey@ni`

See Also

MSAdata-class `check_data()` Dmodel-class Dstock-class Dfishery-class Dsurvey-class DCKMR-class Dtag-class

Examples

```
# Set natal_rs matrix so that the spawning output of stock 1 is
# calculated from mature animals present in regions 1, 2.
# Similarly for stock 2, spawning output from areas 2 and 3.
nr <- 4
ns <- 2
natal_rs <- matrix(0, nr, ns)
natal_rs[1:2, 1] <- natal_rs[2:3, 2] <- 1

# Aggregate stock composition for ages 1-4 and 5-10 across all fleets
na <- 10
na_SC <- 2
SC_aa <- matrix(0, na_SC, na) # Assumes dim(SC_ymafrs)[3] = na_SC
SC_aa[1, 1:4] <- SC_aa[2, 5:10] <- 1

nf <- 3
nf_SC <- 1
SC_ff <- matrix(1, nf_SC, nf) # Assumes dim(SC_ymafrs)[4] <- nf_SC
```

plot-MSA-data *Plotting functions for data in MSA model*

Description

A set of functions to plot data variables and predicted values (catch, age composition, etc.)

Usage

```
plot_catch(fit, f = 1, by = c("region", "stock"), prop = FALSE, annual = FALSE)

plot_index(fit, i = 1, zoom = FALSE)

plot_CAA(fit, f = 1, r = 1, do_mean = FALSE)

plot_CAL(fit, f = 1, r = 1, do_mean = FALSE)

plot_IAA(fit, i = 1, do_mean = FALSE)

plot_IAL(fit, i = 1, do_mean = FALSE)

plot_SC(fit, ff = 1, aa = 1, r = 1, prop = FALSE)

plot_tagmov(fit, s = 1, yy = 1, aa = 1)
```

Arguments

fit	MSAassess object returned by fit_MSA()
f	Integer, indexes the fleet
by	Character to indicate dimension for multivariate plots
prop	Logical, whether to plot proportions (TRUE) or absolute numbers
annual	Logical, whether to plot annual values (summed over seasons)
i	Integer, indexes the survey
zoom	Logical, for plot_index(). If TRUE, plots a subset of years with observed data points. Otherwise, plots predicted values over all model years.
r	Integer, indexes the region
do_mean	Logical, whether to plot full compositions or time series of mean length or mean age
ff	Integer, indexes the aggregate fleet (for stock composition data)
aa	Integer, indexes the aggregate age class (for stock composition and tag data)
s	Integer, indexes the stock
yy	Integer, indexes the aggregate years (for the tag data)

Details

- plot_catch plots the fishery catch by stock or region (either whole numbers or proportions)
- plot_index plots indices of abundance
- plot_CAA plots the fishery catch at age
- plot_CAL plots the catch at length
- plot_IAA plots the index age composition
- plot_IAL plots the index length composition
- plot_SC plots the stock composition
- plot_tagmov plots the tag movements

Value

Various base graphics plots

---

plot-MSA-state *Plotting functions for fitted MSA model*

Description

A set of functions to plot state variables (biomass, recruitment time series, etc.)

Usage

```
plot_S(fit, by = c("stock", "region"), r, s, prop = FALSE, facet_free = FALSE)

plot_B(fit, by = c("stock", "region"), r, s, prop = FALSE, facet_free = FALSE)

plot_R(fit, s)

plot_SRR(fit, s = 1, phi = TRUE)

plot_Rdev(fit, s = 1, log = TRUE)

plot_Fstock(fit, s, by = c("annual", "season"))

plot_self(fit, f = 1, type = c("length", "age"))

plot_seli(fit, i = 1)

plot_selstock(
  fit,
```

```
  s = 1,
  by = c("annual", "season"),
  plot2d = c("contour", "filled.contour"),
  ...
)

plot_N(fit, m = 1, r, s = 1, plot2d = c("contour", "filled.contour"), ...)

plot_V(fit, f = 1, by = c("stock", "region"), prop = FALSE, facet_free = FALSE)

plot_Ffleet(fit, f = 1)

plot_mov(fit, s = 1, y, a, palette = "Peach")

plot_recdist(fit, palette = "Peach")
```

Arguments

fit	MSAassess object returned by fit_MSA()
by	Character to indicate whether to calculate selectivity from F per year or per season
r	Integer for the corresponding region
s	Integer for the corresponding stock
prop	Logical, whether to plot proportions (TRUE) or absolute numbers
facet_free	Logical, whether to allow the y-axis limits to vary by panel in facetted plots
phi	Logical, whether to plot unfished replacement line
log	Logical, whether to plot the natural logarithm of the response variable
f	Integer for the corresponding fleet
type	For plot_self, indicates whether to plot the selectivity by age or length.
i	Integer for the corresponding survey
plot2d	Character, plotting function for either a contour() or filled.contour() plot
...	Other argument to the base graphics function
m	Integer for the corresponding season
y	Integer, year for plotting the movement matrix (last model year is the default)
a	Integer, corresponding age for plotting the movement matrix (age 1 is the default)
palette	Character, palette name to pass to grDevices::hcl.colors(). See grDevices::hcl.pals() for options.

Details

- plot_S plots spawning output by stock or region (either whole numbers or proportions for the latter)

- plot_B plots total biomass by stock or region (either whole numbers or proportions for the latter)
- plot_R plots recruitment by stock
- plot_SRR plots the stock-recruitment relationship and history (realized recruitment) by stock
- plot_Rdev plots recruitment deviations by stock
- plot_Fstock plots apical instantaneous fishing mortality (per year or per season) by stock
- plot_self plots fishery selectivity
- plot_seli plots index selectivity
- plot_selstock plots the realized selectivity from total catch and total abundance at age
- plot_N reports total abundance at age
- plot_V plots vulnerable biomass, availability to the fishery
- plot_Ffleet plots apical instantaneous fishing mortality (per season) by fleet
- plot_mov plots movement matrices and the corresponding equilibrium distribution in multi-area models
- plot_recdist plots the distribution of recruitment for each stock

Value

Various base graphics plots

---

posfun *Quadratic penalty function*

Description

Taped penalty function if x < eps

Usage

```
posfun(x, eps)
```

Arguments

x	Numeric, the parameter
eps	Numeric, the threshold below which a penalty will be applied

Value

The penalty value is

$$\text{penalty} = \begin{cases} 0.1(x-\varepsilon)^2 & x \leq \varepsilon \\ 0 & x > \varepsilon \end{cases}$$

Numeric

`prior` *Priors for MSA model*

Description

Priors in MSA are set by providing character strings which are then parsed into an expression and evaluated in the model environment (see example). This provides flexibility to set a prior for any desired model parameter or variable. See list of parameters in the documentation for `[check_parameters()]` for options (note that priors for `log_rdev_ys` and `log_initrdev_as` are not needed as they're hard-coded into the model). Several functions below generate the character string for the prior for important dynamics parameters, such as natural mortality and steepness.

Usage

```
prior_h(MSAdata, s = 1, m, stdev)

prior_M(MSAdata, s = 1, meanlog, sdlog)

prior_q(MSAdata, i = 1, meanlog, sdlog)
```

Arguments

`MSAdata`	Data object. Class MSAdata
`s`	Integer for stock
`m`	Mean in un-transformed space
`stdev`	Standard deviation in un-transformed space
`meanlog`	Mean of the lognormal distribution on the log scale
`sdlog`	Standard of the lognormal distribution on the log scale
`i`	Integer for the corresponding index

Details

- `prior_h` returns the log prior for stock-recruit steepness. Beta distribution for the Beverton-Holt function and normal distribution for Ricker function.
- `prior_M` returns the log prior for natural mortality. Lognormal distribution.
- `prior_q` returns the log prior for index catchability. Lognormal distribution.

Value

Character.

Examples

```
# Add M and steepness prior to model

dat <- new("MSAdata")
dat@Dmodel@ns <- 1
dat@Dstock@SRR_s <- "BH"

pr_M <- prior_M(dat, s = 1, log(0.25), 0.3)
pr_h <- prior_h(dat, s = 1, 0.8, 0.15)
dat@Dmodel@prior <- c(pr_M, pr_h)
```

---

profile *Profile parameters of MSA model*

Description

Evaluate change in objective function and likelihood components for up to 2 parameters.

Usage

```
## S4 method for signature 'MSAassess'
profile(fitted, p1, v1, p2, v2, cores = 1, ...)

## S4 method for signature 'MSAassess'
plot(
  x,
  component = "objective",
  rel = TRUE,
  xlab,
  ylab,
  main,
  plot2d = c("contour", "filled.contour"),
  ...
)
```

Arguments

fitted	MSAassess object returned by fit_MSA()
p1	Character string that represents the first parameter to be profiled, including the parameter name and index of the vector/array. See "Parameters" section of make_parameters(). Additionally, this function allows users to specify R0_s and h_s (in normal units).
v1	Vector of values corresponding to p1

p2	Character string that represents the optional second parameter to be profiled
v2	Vector of values corresponding to p2
cores	Integer for the number of cores to use for parallel processing (snowfall package)
...	Other argument to the base graphics function, i.e., either plot() or contour()
x	Output from profile.MSAassess()
component	Character for the column in x to be plotted
rel	Logical, whether the relative change in component is plotted (TRUE) or the raw values (FALSE)
xlab	Optional character for the x-axis label
ylab	Optional character for the y-axis label
main	Optional character for the plot title
plot2d	Character, plotting function for two-dimensional profiling (either a contour() or filled.contour() plot)

Value

The profile generic returns a data frame of the likelihood values that correspond to fixed values of p1 and p2.

- Likelihood loglike refers to maximizing the probability of the observed data (higher values for better fit)
- Prior logprior refers to maximizing the probability of a parameter to their prior distribution (higher values are closer to the prior mode)
- Penalty penalty are values added to the objective function when parameters exceed model bounds (lower values are better)
- fn is the objective function returned by RTMB (lower values are better)
- objective is the objective function returned by the optimizer (lower values are better)

The accompanying plot function returns a line plot for a 1-dimensional profile or a contour plot for a two dimensional profile. Will plot the negative log likelihood or negative log prior (better fit with lower values).

Relative values are obtained by subtracting from the fitted value. See attr(x, "fitted")

---

report *Generate markdown reports*

Description

Generate a markdown report of model fits and estimates.

Usage

```
report(object, ...)

## S4 method for signature 'MSAassess'
report(
  object,
  name,
  filename = "MSA",
  dir = tempdir(),
  open_file = TRUE,
  render_args = list(),
  ...
)
```

Arguments

object	An object from MSA.
...	Additional arguments to render reports.
name	Optional character string for the model name to include in the report, e.g., model run number. Default uses substitute(object)
filename	Character string for the name of the markdown and HTML files.
dir	The directory in which the markdown and HTML files will be saved.
open_file	Logical, whether the HTML document is opened after it is rendered.
render_args	List of arguments to pass to rmarkdown::render().

Value

report generic for MSAassess invisibly returns the output of rmarkdown::render(): character of the path of the rendered HTML markdown report.

---

residuals *Calculate model residuals*

Description

Extract residuals from fitted model

Usage

```
## S4 method for signature 'MSAassess'
residuals(object, vars, type = c("response", "pearson"), ...)
```

Arguments

`object`	MSAassess object returned by `fit_MSA()`
`vars`	Character vector to indicate which residuals will be calculated. Available choices from MSAdata object are: "Cinit_mfr", "Cobs_ymfr", "CAAobs_ymafr", "CALobs_ymlfr", "Iobs_ymi", "IAAobs_ymai", "IALobs_ymli", "SC_ymafrs"
`type`	Character, 'response' for the `log(observed/predicted)` values or 'pearson' for calculating Z-scores. Composition data always use 'pearson'.
`...`	Not used

Value

A named list based on `vars` argument.

`retrospective` *Retrospective analysis*

Description

Perform a retrospective analysis, successive removals of most recent years of data to evaluate consistency in model estimates of biomass, recruitment, etc.

The `summary` method returns Mohn's rho and the plot method generates a markdown report.

Usage

```
retrospective(MSAassess, yret = 0:5, cores = 1)

## S3 method for class 'MSAretro'
plot(
  x,
  var = c("S_yst", "R_yst", "F_yst", "log_rdev_yst", "VB_ymft"),
  s = 1,
  f = 1,
  ...
)

## S3 method for class 'MSAretro'
summary(object, by = c("stock", "fleet"), ...)

## S3 method for class 'MSAretro'
report(
  object,
  filename = "retro",
  dir = tempdir(),
  open_file = TRUE,
  render_args = list(),
  ...
)
```

Arguments

MSAassess	MSAassess object
yret	Vector specifying the years (positive integers and include zero) to remove for the retrospective analysis
cores	Integer for the number of cores to use for parallel processing (snowfall package)
x, object	Output of retrospective function
var	Character to indicate the metric, the item in the MSAretro list to be plotted. See details below.
s	Integer for the stock index to plot
f	Integer for the fleet index to plot
...	Not used
by	Character indicating whether to calculate to Mohn's rho on stock or fleet-based time series
filename	Character string for the name of the markdown and HTML files.
dir	The directory in which the markdown and HTML files will be saved.
open_file	Logical, whether the HTML document is opened after it is rendered.
render_args	List of arguments to pass to rmarkdown::render().

Value

A MSAretro object containing a named lists of arrays generated by the retrospective analysis:

- S_yst Spawning output array [y, s, t] where t indexes the retrospective peel
- R_yst Recruitment array [y, s, t]
- F_yst Apical fishing mortality [y, s, t]
- VB_ymft Vulnerable biomass available to each fishery [y, m, f, t]

plot generic for MSAretro returns individual figures using base graphics.

summary generic for MSAretro returns a matrix of Mohn's rho.

report generic for MSAretro invisibly returns the output of rmarkdown::render(): character of the path of the rendered HTML markdown report.

---

simulate *Simulate data*

Description

Simulate data observations from fitted MSA model.

Usage

```
## S4 method for signature 'MSAassess'
simulate(object, nsim = 1, seed = NULL, ...)
```

Arguments

`object`	MSAassess object returned by `fit_MSA()`
`nsim`	Integer, number of simulations
`seed`	Random number generator seed
`...`	Not used

Value

A list of `nsim` length with data observations

`softmax` *Softmax function*

Description

Takes a vector of real numbers and returns the corresponding vector of probabilities

Usage

```
softmax(eta, log = FALSE)
```

Arguments

`eta`	Vector
`log`	Logical, whether to return the value of the logarithm

Details

Uses `multiSA:::logspace.add` for numerical stability

Value

Numeric, vector length of `eta`: $\exp(\eta)/\sum\exp(\eta)$

summary *Summarize MSA model output*

Description

Report parameter estimates, gradients, and standard errors

Usage

```
## S4 method for signature 'MSAassess'
summary(object, ...)
```

Arguments

object	Output from MSA
...	Not used

Value

Matrix

Index

∗ **MSAassess**
MSAassess-class, 39
∗ **MSAdata**
MSAdata-class, 39

calc_eqdist, 3
calc_F, 3
calc_F(), *15, 16*, *38*
calc_fsel_age *(*conv_selpar*)*, 20
calc_fsel_age(), *20, 21*
calc_growth, 6
calc_growth(), *28*, *41*
calc_HSP *(*calc_POP*)*, 12
calc_HSP(), *34*
calc_index, 7
calc_isel_age *(*conv_selpar*)*, 20
calc_isel_age(), *20, 21*
calc_LAK, 8
calc_LAK(), *28*, *41*
calc_nextN, 9
calc_NPR *(*calc_phi_simple*)*, 11
calc_NPR(), *12*
calc_phi_project, 10
calc_phi_project(), *12*
calc_phi_simple, 11
calc_phi_simple(), *11, 12*
calc_POP, 12
calc_POP(), *34*
calc_population, 14
calc_population(), *10, 11*
calc_recruitment, 16
calc_recruitment(), *15*, *28*, *41*
calc_sel_len *(*conv_selpar*)*, 20
calc_sel_len(), *20, 21*
check_data, 17
check_data(), *25*, *27, 28*, *30, 31*, *41*, *44*
check_parameters *(*make_parameters*)*, 36
check_parameters(), *31*, *36, 37*
CondExpEq *(*CondExpLt*)*, 17
CondExpGe *(*CondExpLt*)*, 17
CondExpGt *(*CondExpLt*)*, 17
CondExpLe *(*CondExpLt*)*, 17
CondExpLt, 17
contour(), *47*, *51*
conv_mov, 19
conv_mov(), *38*
conv_selpar, 20
conv_selpar(), *20, 21*, *30*, *38*, *43*
conv_Sigma, 22

DCKMR, *40*
DCKMR-class, 23, *25*, *27, 28*, *30, 31*, *44*
Dfishery, *40*
Dfishery-class, 23, *25*, *27, 28*, *30, 31*, *44*
Dlabel, *40*
Dlabel-class, 25
Dmodel, *40*
Dmodel-class, *25*, 26, *27, 28*, *30, 31*, *44*
Dstock, *11*, *15*, *40*
Dstock-class, *25*, *27*, 27, *28*, *30, 31*, *44*
Dsurvey, *40*
Dsurvey-class, *25*, *27, 28*, 29, *30, 31*, *44*
Dtag, *40*
Dtag-class, *25*, *27, 28*, *30*, 30, *31*, *44*

filled.contour(), *47*, *51*
fit_MSA, 31
fit_MSA(), *45*, *47*, *50*, *55*

get_MSAdata, 32
get_sdreport, 33
grDevices::hcl.colors(), *47*
grDevices::hcl.pals(), *47*

like_CKMR, 34
like_CKMR(), *13*, *23*, *43*
like_comp, 34
like_comp(), *24*, *29, 30*, *42*, *44*

make_map *(*make_parameters*)*, 36
make_map(), *37*

make_parameters, 36
make_parameters(), *26*, *31*, *36–38*, *40*, *50*
MSAassess, *32*, *45*, *47*, *50*, *53–55*
MSAassess-class, 39
MSAdata, *6*, *17*, *31*, *32*, *37*, *39*, *49*, *53*
MSAdata-class, *25*, *27*, *28*, *30*, *31*, 39, *44*

plot,MSAassess-method *(*profile*)*, 50
plot-MSA-data, 45
plot-MSA-state, 46
plot.MSAprof *(*profile*)*, 50
plot.MSAretro *(*retrospective*)*, 53
plot_B *(*plot-MSA-state*)*, 46
plot_CAA *(*plot-MSA-data*)*, 45
plot_CAL *(*plot-MSA-data*)*, 45
plot_catch *(*plot-MSA-data*)*, 45
plot_Ffleet *(*plot-MSA-state*)*, 46
plot_Fstock *(*plot-MSA-state*)*, 46
plot_IAA *(*plot-MSA-data*)*, 45
plot_IAL *(*plot-MSA-data*)*, 45
plot_index *(*plot-MSA-data*)*, 45
plot_mov *(*plot-MSA-state*)*, 46
plot_N *(*plot-MSA-state*)*, 46
plot_R *(*plot-MSA-state*)*, 46
plot_Rdev *(*plot-MSA-state*)*, 46
plot_recdist *(*plot-MSA-state*)*, 46
plot_S *(*plot-MSA-state*)*, 46
plot_SC *(*plot-MSA-data*)*, 45
plot_self *(*plot-MSA-state*)*, 46
plot_seli *(*plot-MSA-state*)*, 46
plot_selstock *(*plot-MSA-state*)*, 46
plot_SRR *(*plot-MSA-state*)*, 46
plot_tagmov *(*plot-MSA-data*)*, 45
plot_V *(*plot-MSA-state*)*, 46
posfun, 48
posfun(), *5*
prior, *27*, *41*, 49
prior_h *(*prior*)*, 49
prior_M *(*prior*)*, 49
prior_q *(*prior*)*, 49
profile, 50
profile,MSAassess-method *(*profile*)*, 50
profile.MSAassess *(*profile*)*, 50
profile.MSAassess(), *51*

report, 51
report(), *32*
report,MSAassess-method *(*report*)*, 51
report.MSAassess *(*report*)*, 51
report.MSAretro *(*retrospective*)*, 53
residuals, 52
residuals,MSAassess-method *(*residuals*)*, 52
retrospective, 53
retrospective(), *32*
rmarkdown::render(), *52*, *54*
RTMB::MakeADFun(), *31–33*, *39*
RTMB::sdreport(), *32*, *33*, *39*

simulate, 54
simulate,MSAassess-method *(*simulate*)*, 54
simulate.MSAassess *(*simulate*)*, 54
softmax, 55
stats::nlminb(), *31*, *32*, *39*
summary, 56
summary,MSAassess-method *(*summary*)*, 56
summary.MSAretro *(*retrospective*)*, 53

TMB::MakeADFun(), *32*, *36*