Package 'MixFishSim'

August 16, 2017

Title Mixed Fishery fleet dynamics simulation tool

Version 0.0.0.9000

Description A simulation framework for evaluating fleet dynamics in mixed fisheries.

Depends R (>= 3.3.1),

Imports spate

License What license is it under?

Encoding UTF-8

LazyData true

RoxygenNote 5.0.1

Suggests testthat

R topics documented:

---

baranov_f | *Baranov F*

Description

baranov_f provides the function to solve in find_f for estimating weekly fishing mortality from catch (C), biomass (B) and natural mortality (M). It's based on the standard Baranov catch equation.

1

Usage

```
baranov_f(F, C, B, M)
```

Arguments

F	is the fishing mortality rate to solve.
C	is a Numeric vector detailing the catch at wk_t
B	is a Numeric vector of the biomass at wk_t
M	is a Numeric vector of the natural mortality rate at wk_t

Value

returns nothing, is objective to be solved by [find_f](#)

Examples

```
## No examples
```

---

create_fields   *Create species distribution fields*

Description

create_fields parametrises and returns the spatio-temporal fields used for the spatial distribution of fish populations and movement in space and time for the simulations.

The spatio-temporal fields are generated using [spate.sim](#) function from the *spate* package using an advective-diffusion Stochastic Partial Differential Equation (SPDE). See *Lindgren 2011 and Sigrist 2015* for further detail.

Usage

```
create_fields(npt = 1000, t = 1, seed = 123, n.spp = NULL,
  spp.ctrl = NULL, plot.dist = FALSE, plot.file = getwd())
```

Arguments

npt	Numeric integer with the dimensions of the field in *npt * npt*
t	Numeric integer with the number of time-steps in the simulation
seed	(Optional) Numeric integer with the seed for the simulation
n.spp	Numeric integer with the number of species to be simulated. Each species must have an individual control list as detailed below.
spp.ctrl	List of controls to generate each species spatio-temporal distribution. Must be of the form spp.ctrl = list(spp.1 = c(rho0 = 0.001, ...), spp.2 = c(rho0 = 0.001, ..),..) and contain the following:

- **rho0** (*>=0*) Controls the range in a matern covariance structure.
- **sigma2** (*>=0*) Controls the marginal variance (i.e. process error) in the matern (*>=0*) covariance structure.
- **zeta** (*>=0*) Damping parameter; regulates the temporal correlation.

- **rho1** (*>=0*) Range parameter for the diffusion process
- **gamma** (*>=0*) Controls the level of anisotropy
- **alpha** (*[0, $\pi/2$]*) Controls the direction of anisotropy
- **muX** (*[-0.5, 0.5]*) x component of drift effect
- **muY** (*[-0.5, 0.5]*) y component of drift effect
- **tau2** (*>=0*) Nugget effect (measurement error)
- **nu** Smoothness parameter for the matern covariance function

plot.dist	Boolean, whether to plot the distributions to file
plot.file	path to save the plots of the species distributions

Value

Silently returns a list of spatial distributions with first level of the list being the population (1 -> n.spp) and the second being time (1 -> t). If `plot.dist = TRUE` it produces an image of the spatial distributions at each time step for each of the populations saved to the working directory (unless specified otherwise in `plot.file`)

Examples

```
fields <- create_fields(n.spp = 1, t = 2,
     spp.ctrl = list(
     'spp.1' = c('rho0' = 0.1, 'sigma2' = 1, 'zeta' = 0.1,
  'rho1' = 0.01, 'gamma' = 0.3, 'alpha' = pi/4,
  'muX' = -0.05, 'muY' = -0.05, 'tau2' = 0, 'nu' = 1.5)),
plot.dist = TRUE, plot.file = getwd())
```

---

create_hab	*Create habitat distribution fields*

Description

`create_hab` parametrises and returns the spatial fields used for the distribution of suitable habitat for the populations in the simulation.

The spatial fields are generated using RFsimulate function from the *RandomFields* package.

Usage

```
create_hab(npt = 100, seed = 123, n.spp = NULL, spp.ctrl = NULL,
  plot.dist = FALSE, plot.file = getwd())
```

Arguments

npt	Numeric integer with the dimensions of the field in *npt * npt*
n.spp	Numeric integer with the number of species to be simulated. Each species must have an individual control list as detailed below.
spp.ctrl	List of controls to generate suitable habitat for each species. Must be of the form spp.ctrl = list(spp.1 = c(var = 20, ...), spp.2 = c(var = 10, ..),..) and contain the following:

- **nu** (*>=0*)

- **var** (>=*0*) Controls the range in a matern covariance
- **scale** (>=*0*)
- **Aniso** (*matrix, dim = c(2,2)*)

plot.dist	Boolean, whether to plot the distributions to file
plot.file	path to save the plots of the species distributions

Value

Silently returns a list of spatial distributions of suitable habitat with first level of the list being the population (1 -> n.spp). If plot.dist = TRUE it produces an image of the spatial distributions at each time step for each of the populations saved to the working directory (unless specified otherwise in plot.file)

Examples

```
fields <- create_hab(n.spp = 1,
    spp.ctrl = list(
    'spp.1' = list('nu' = 1/0.15, var = 1, scale = 10, Aniso =
    matrix(nc=2, c(1.5, 3, -3, 4)))), plot.dist = TRUE, plot.file =
    getwd())
```

---

delay_diff	*Delay-difference (weekly)*

Description

delay_difference implements a two-stage delay-difference model with a weekly time-step after *Dichmont 2003*. Given the starting biomass, overall mortality and recruitment it returns the biomass in wk+1.

Usage

```
delay_diff(K = 0.3, F = NULL, M = 0.2, wt = 1, wtm1 = 0.1, R = NULL,
  B = NULL, Bm1 = NULL, al = NULL, alm1 = NULL)
```

Arguments

K	is a Numeric vector describing growth @param F is the weekly. Note: K is transformed to rho with $\rho = exp-K$ for the model. estimate of instantaneous fishing mortality (obtained elsewhere, via find_f and baranov_f functions.
M	is a Numeric vector of the instantaneous rate of natural mortality for the population
wt	is a Numeric vector of the weight of a fish when fully recruited
wtm1	is a Numeric vector of the weight of a fish before its recruited
R	is a Numeric vector of the annual recruitment for the population in numbers
B	is the biomass of the population during wk_t
Bm1	is a Numeric vector of the biomass of the population in the previous week wk_{t-1}
al	is a Numeric vector of the proportion of recruits to the fishery in wk_t
alm1	is a Numeric vector of the proportion of recruits to the fishery in wk_{t-1}

Value

Returns the biomass at the beginning of the following week, wk_{t+1}

Examples

```
delay_diff(K = 0.3, F = 0.2, M = 0.2, wt = 1, wtm1 = 0.1, R = 1e6, B = 1e5,
Bm1 = 1e4, al = 0.5, alm1 = 0.1)
```

---

find_f	*find F (fishing mortality)*

Description

find_f uses uniroot to find the fishing mortality rate given the catch, biomass and natural mortality using the baranov_f objective function.

Usage

```
find_f(C = C, B = B, M = M, FUN = baronov_f)
```

Arguments

C	is a Numeric vector detailing the catch at wk_t
B	is a Numeric vector of the biomass at wk_t
M	is a Numeic vector of the natural mortality rate at wk_t
FUN	is the objective function, here the Baranov equation baranov_f

Value

Gives the fishing mortality estimate *F*

Examples

```
find_f(C = 3000, B = 12000, M = 0.2, FUN = baranov_f)
```

---

init_sim	*Initialise simulation*

Description

init_sim sets up the general simulation parameters such as number of tows in a day, number of days fished in a week, how often species movement occurs and number of years for the simulation. It also creates some vector and matrix structures which are used in the init_pop and init_fleet functions.

Usage

```
init_sim(n_years = 1, n_tows_day = 4, n_days_wk_fished = 5,
  n_fleets = 1, n_vessels = 1, n_species = 1, move_length = 2)
```

Arguments

n_years	is an integar defining the number of years for the simulation
n_days_wk_fished	
	is an integar defining the number of days in a calendar week that are fished (e.g. 5 (out of 7))
n_fleets	is an integar defining the number of fleets in the simulation
n_vessels	is an integar defining the number of vessels in each fleet
n_species	is an integar defining the number of species in the simulation
move_length	is an integar defining the duration (in weeks) between spatial movements for the populations
n_tow_day	is an integar defining the number of tows in a days fishing

Value

is a list of lists, detailing the indexs and data formats necessary for the simulation.

Examples

```
init_sim(n_years = 1, n_tows_day = 4, n_days_wk_fished = 5,
n_fleets = 1, n_vessels = 1, n_species = 1, move_length = 2)
```

---

make_step	*make step function*

Description

make_step determines the new position of the vessel following a move, using the step distance and bearing inputs.

Usage

```
make_step(stepD, Bear, start.x, start.y)
```

Arguments

stepD	is a Numeric vector of the distance to move
Bear	is a Numeric vector of the bearing to move (in degrees)
start.x	is the starting point on the x-axis
start.y	is the starting point on the y-axis

Value

returns a new coordinate position through a vector (x, y)

Examples

```
make_step(stepD = 20, Bear = 90, start.x = 20, start.y = 5)
```

Recr   *Recruitment function*

Description

`Recr` returns a biomass of recruited fish to the population based on a stock-recruit relationship and some measure of variation.

Usage

```
Recr(model, params, B, cv, ..)
```

Arguments

model	is a character detailing the recruitment function to use (currently 'BH' for Beverton and Holt or 'Ricker' for a Ricker stock-recruit relationship.
params	is a Numeric vector of length 2, containing labelled *a* and *b* parameters for the stock-recruit function. For Beverton and Holt *a* refers to the maximum recruitment rate in biomass, *b* refers to the Spawning Stock Biomass (SSB) required to produce half the maximum. For Ricker *a* refers to the maximum productivity per spawner and *b* the density dependent reduction in productivity as the stock increases.
B	is a Numeric vector containing the SSB of the adult population from which the recruitment derives.
cv	is a Numeric vector containing the coefficient of variation in the recruitment function.

Value

returns the recruitment to the population in biomass.

Examples

```
Recr(model = 'BH', params = c("a" = 2000, "b" = 200), B = 1000, cv = 0.1)
```

---

step_length   *Step length function*

Description

`step_length` is a function to calculate the step length a vessel takes based on the step parameters provided for a gamma function and the revenue from the most recent fishing activity.

Usage

```
step_length(step_params = params[["step_params"]], revenue = revenue)
```

Arguments

step_params   is a list of parameters which determine the relationship between revenue gained
               from the recent fishing activity and the next move step length, based on a gamma
               function. The list contains the following:

                   • **rate** Determines the rate ....
                   • **B1** Determines...
                   • **B2** Determines ...
                   • **B3** Determines ..

revenue   is the last observed fishing revenue for the vessel

Value

step - the size of the next step

Examples

```
step_length(step_params = list(B1 = 1, B2 = 50, B3 = 2000, rate = 1),
revenue = 300)
```

---

  test_step   *test step length function*

Description

`test_step` is a function to test and review parameters for the step_length function. This is primarily
to help with identifying the right parameters for the desired relationship between revenue and step
length.

Usage

```
test_step(step_params = step_params, rev.max = 2000)
```

Arguments

step_params   is a list of parameters which determine the relationship between revenue gained
               from the recent fishing activity and the next move step length, based on a gamma
               function. The list contains the following:

                   • **rate** Determines the rate ....
                   • **B1** Determines...
                   • **B2** Determines ...
                   • **B3** Determines ..

rev.max   is the maximum revenue at which to test the step length function.

Value

is a plot of the relationship between revenue and step length

Examples

```
test_step(step_params = list(B1 = 1, B2 = 50, B3 = 2000, rate = 1), rev.max
= 2000)
```

Index