Package 'MixFishSim'

September 12, 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 dplyr,
RandomFields,
Rcpp

License What license is it under?

Encoding UTF-8

LazyData true

RoxygenNote 5.0.1

Suggests testthat

LinkingTo Rcpp

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.

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

Arguments

`nrows`	Numeric integer with the y dimension of the field in *nrow * ncol*
`ncols`	Numeric integer with the x dimension of the field in *nrow * ncol*
`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(nrows = 100, ncols = 100, 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())
```

`create_spawn_hab` *create spawning habitat*

Description

`create_spawn_hab` modifies the habitat preference maps created by `create_hab` to account for spawning habitat preference - can be used as a substitute during spawning periods.

Usage

```
create_spawn_hab(hab = hab, spwnareas = NULL, mult = 10)
```

Arguments

hab	is the habitat preference for the population
spwnareas	is a list of Numeric vectors with the West, East, South and North dimensions of the spawning areas, in the form list(spwn1 = c(x1, x2, y1, y2)
mult	is a Numeric with the attractiveness of the spawning area (a multiplier)

Value

is the new habitat preference, taking account of the spawning area

Examples

```
create_spawn_hab(hab = matrix(nc = 100, runif(100 *
100)), spwnareas = list(spwn1 = c(20, 30, 50, 60)), mult = 10)
```

---

define_spawn *define spawning areas*

Description

`define_spawn` is an auxiliary function called by `create_spawn_hab` to create the spawning habitat preferences.

Usage

```
define_spawn(coord = NULL, spwn = NULL, mult = 10)
```

Arguments

coord	is a List of Numeric vectors of the boundaries of the spawning areas, i.e. list(spwn1 = c(x1, x2, y1, y2), spwn2 = ...)
spwn	is a Numeric matrix of 1s fed in by `create_spawn_hab`
mult	is a Numeric of the attractiveness of the spawning areas

Value

a matrix of spawning preference

Examples

```
define_spawn(coord = list(spwn1 = c(2,4,2,4)), spwn = matrix(nc = 3, runif(9)), mult = 10)
```

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)
```

`distance_calc` *distance calculation*

Description

`distance_calc` calculates the euclidean distance between two cell references.

Usage

```
distance_calc(x1, y1, x2, y2)
```

Arguments

`x1`	is an integar for the starting x position
`y1`	is an integar for the starting y position
`x2`	is an integar for the end x position
`y2`	is an integar for the end y position

Value

is a distance between the two cells

Examples

```
distance_calc(2, 3, 5, 7)
```

`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)
```

`fish_iters` *Go fish iter*

Description

`go_fish_iter` applies the function `go_fish` to the entire fleet an lapply.

Usage

```
fish_iters(FUN, N = 1, fleets_params = NULL, fleets_catches = NULL,
  Pop = NULL, t = t, ...)
```

Arguments

`fleets_params`	is the parameter settings initialised from `_init_fleets`
`fleets_catches`	is the DF initialised from `_init_fleets`
`Pop`	is the population matrix for all populations

Value

is a list with the objects catch detailing the fleet catches and catch_matrix detailing the spatial catches, to input to the delay difference model

Examples

```
None as yet
```

`go_fish` *Go fish*

Description

`go_fish` is a function used to apply the fishing simulation model

Usage

```
go_fish(sim_init = sim, fleet_params = NULL, fleet_catches = NULL,
  pops = Pop, t = t)
```

Arguments

`sim_init`	is the initialised object from `init_sim`.
`fleet_params`	is the parameter settings initialised from `_init_fleets`
`fleet_catches`	is the DF initialised from `_init_fleets`

Value

is ...

init_fleet *Initialise fleet*

Description

`init_fleet` sets up the parameters and results data frame to record the catches from the simulation.

Usage

```
init_fleet(sim_init = NULL, n_fleets = 1, n_vessels = 1, VPT = NULL,
  Qs = NULL, step_params = NULL, past_knowledge = FALSE,
  past_year_month = FALSE, past_trip = FALSE, threshold = NULL)
```

Arguments

`sim_init`	is the output (a list) from the `sim_init` function with the indexing for the simulation.
`n_fleets`	is an integer of the number of fleets in the model
`n_vessels`	is an integer of the number of vessels in each fleet
`VPT`	is a named vector of numerics detailing the value-per-tonne for catches from each of the species (same for all fleets)
`Qs`	is a list (an element for each fleet) with each element containing a named vector with the catchability parameters for each species the vessels in the fleet
`step_params`	is a list (an element for each fleet) with each element containing a named vector with the step parameters used in `step_length`. This must include the named elements **rate**, **B1**, **B2**, **B3**.
`past_knowledge`	is a Boolean (TRUE / FALSE) whether past knowledge should determine fishing location (only after the first year)
`past_year_month`	is a Boolean (TRUE / FALSE) that indicates whether the same month in previous years should be included in the past knowledge decision
`past_trip`	is a Boolean (TRUE / FALSE) that indicates whether the past trip undertaken should be included in the past knowledge decision
`knowledge_threshold`	is a numeric (0 - 1) detailing the threshold at which a fishing tow should be considered "good" and included in the selection of possible choices of starting fishing locations in future tows.

Value

is a list with two elements containing the fleet parameters, a named list **fleet_params**, and the fleet catches, **catches_list**, which is a list of a list. For the**catches_list** the first element denotes the fleet number, the second element is the vessel number with a dataframe for recording the vessels catches.

Examples

```
None yet, to add
```

init_pop *Initialise populations*

Description

init_pop sets up the populations spatial distribution based on the habitat preference, starting cell and 'n' numbers of movements for all populations in the simulation.

Usage

```
init_pop(sim_init = sim_init, Bio = NULL, hab = NULL, start_cell = NULL,
  lambda = NULL, init_move_steps = 10, rec_params = NULL, rec_wk = NULL,
  spwn_wk = NULL, M = NULL)
```

Arguments

Bio	is a named Numeric vector of the starting (total) biomass for each of the populations.
hab	is the list of Matrices with the habitat preferences created by create_hab
start_cell	is a list of Numeric vectors with the starting cells for the populations
lambda	is the strength that the movement distance decays at in the move_prob function
init_move_steps	is a Numeric indicating the number of movements to initialise for the population distributions
rec_params	is a list with an element for each population, containing a vector of the stock recruit parameters which must contain **model**, **a**, **b** and **cv**. See Recr for details.
rec_wk	is a list with an element for each population, containing a vector of the weeks in which recruitment takes place for the population
spwn_wk	is a list with an element for each population, containing a vector of the weeks in which spawning takes place for the population
M	is a named vector, with the annual natural mortality rate for each population

Value

The function returns the recording vectors at the population level, the spatial matrices for the starting population densities and the demographic parameters for each population

Examples

```
init_pop(sim_init = sim_init, Bio = c("spp1" = 1e6, "spp2" = 2e5), hab = list(spp1 = matrix(nc = 10,
runif(10*10)), spp2 = matrix(nc = 10, runif(10*10)), lambda = c("spp1" =
0.2, "spp2" = 0.3), init_move_steps = 10), rec_params = list("spp1" =
c("model" = "BH", "a" = 10, "b" = 50, "cv" = 0.2), "spp2" = c("model" = "BH",
"a" = 1, "b" = 8, "cv" = 0.2)), rec_wk = list("spp1" = 13:16, "spp2" =
13:18, spwn_wk = list("spp1" = 15:18, "spp2" = 18:20),M = c("spp1" = 0.2, "spp2" = 0.1)))
Note, example will not have the right biomass
```

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, nrows = nrows,
  ncols = ncols, move_freq = 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
nrows	Numeric integer with the y dimension of the field in *nrow * ncol*
ncols	Numeric integer with the x dimension of the field in *nrow * ncol*
move_freq	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_freq = 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)
```

`move_population` *population movement function*

Description

`move_population` redistributes the population based on the movement probabilities

Usage

```
move_population(moveProp, StartPop)
```

Arguments

`moveProp`	is a list of the proportion of the population from each cell to reallocated to each of the other cells
`StartPop`	is a Numeric Matrix of the current populations distribution

Value

is a list of the new position for the population from each of the cells.

NOTE: This is not aggregated and requires calling the R function *Reduce('+', Lst)* to reaggregate. Would be better if done in function but Reduce is currently faster...but much more memory intensive to get out the lists...using the standard c++ accumulate function may work for this but untested

Examples

```
None at the moment
```

---

move_prob *movement probability function*

Description

move_prob calculates the movement probability between a cell and all other cells based on the distance and *lambda*.

Usage

```
move_prob(start, lambda, hab)
```

Arguments

start	is a Numeric vector of dim 2 for the starting position c(x,y)
lambda	is an integar for the value for the exponential decay in probability of movement, i.e. $Pr(B\|A) = \exp -\lambda * dist_{a,b}/Sum(c = 1 : c = n) \exp -\lambda * dist$
hab	is a matrix of the habitat suitability

Value

is a matrix of the movement probabilities from a cell

Examples

```
move_prob(c(2, 5), 0.3, matrix(nc = 3, runif(9)))
```

---

move_prob_Lst *movement probability function as a list*

Description

move_prob_list applies move_prob from all cells to all other cells and returns as a list.

Usage

```
move_prob_Lst(lambda, hab)
```

Arguments

lambda	is the decay value as in move_prob
hab	is a matrix of the habitat suitability for the population

Value

is a list of the movement probabilities form each cell to all other cells

Examples

```
None at the moment
```

`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