Package 'LobsterCatch'

January 20, 2025

Title Models the Capture Processes in American Lobster Trap Fishery

Version 0.1.0

Description Simulate lobster catch process in a trap fishery. Factors such as lobster density on ocean floor, their movement, trap saturation and bait shrinkage rate can be modeled. Details of the methods for modeling those processes can be found in: Addison and Bell (1997) <doi:10.1071/MF97169>.

License GPL (>= 3)

Depends R (>= 2.10)

Imports stats, utils

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

NeedsCompilation no

Author Pourfaraj vahab [aut],
Cook Adam [aut],
Jamie Tam [ctb],
Nickchi Payman [aut, cre]

Maintainer Nickchi Payman <payman.nickchi@gmail.com>

Repository CRAN

Date/Publication 2023-06-19 15:40:05 UTC

Contents

`catchability` *This function calculates the probability of entry into a trap, also known as catchability. It includes the parameters described in Addison and Bell (1997), and can also incorporate the length of the catch while calculating the catchability.*

Description

This function calculates the probability of entry into a trap, also known as catchability. It includes the parameters described in Addison and Bell (1997), and can also incorporate the length of the catch while calculating the catchability.

Usage

```
catchability(
  q0,
  qmin,
  saturationThreshold,
  Ct,
  lengthBased,
  lobLengthThreshold,
  lobSize = NA,
  sexBased,
  lobSex
)
```

Arguments

`q0`	is the initial probability of entry into an empty trap (range is from 0-1). Default value is 0.5.
`qmin`	is the asymptotic minimum probability of entry with default value being 0.
`saturationThreshold`	is the number of lobsters in a trap at which the probability of another lobster entering the trap is zero (i.e. no more entry due to agnostic behavior of trapped lobsters).
`Ct`	is the number of caught lobster
`lengthBased`	Logical. If TRUE the length of lobsters caught will be taken into account

`lobLengthThreshold`
: Logical.If TRUE the carapace length (in milliliters) beyond which there is no chance of catching another lobster due to bold agnostic behavior of large lobsters.

`lobSize`
: is a size frequency dataset that is representative of the population and can be incorporated to the model.

`sexBased`
: Logical. If TRUE, lobster sex is taken into account and user must provide a list containing sex distribution for the simulated population

`lobSex`
: is the sex of trapped lobster

Value

Returns the probability of entry to trap.

References

Julian T. Addison and Michael C. Bell (1997), Simulation modelling of capture processes in trap fisheries for clawed lobsters, Marine Freshwater Research, 48(8), 1035-1044, https://www.publish.csiro.au/MF/MF97169

`directionalMove`
: *This function models movement of lobsters toward the trap.The distance of lobsters to trap determines the magnitude of those moves. As lobster gets closer to the trap, the magnitude of its directional move becomes larger and the random move becomes smaller.*

Description

This function models movement of lobsters toward the trap.The distance of lobsters to trap determines the magnitude of those moves. As lobster gets closer to the trap, the magnitude of its directional move becomes larger and the random move becomes smaller.

Usage

```
directionalMove(
  Lobster,
  dStep,
  minDistoTrap,
  Trap,
  radiusOfInfluence,
  currentZoI
)
```

Arguments

`Lobster`	location of lobster in the grid in x and y coordinates.
`dStep`	Distance that each lobster moves during one time step.
`minDistoTrap`	Distance from the trap.
`Trap`	location of trap in the arena.
`radiusOfInfluence`	Radius of influence for the baited trap.
`currentZoI`	Radius of influence in each time step given the bait shrinkage.

Value

Returns the new coordinates of each lobster in the arena after each directional move.

`dispersion` *This function calculates the variance to mean ratio (also known as dispersion index).*

Description

This function calculates the variance to mean ratio (also known as dispersion index).

Usage

```
dispersion(x)
```

Arguments

`x`	is a numeric vector.

Value

Returns the dispersion index.

`distanceToClosestTrap` *The function finds the closest trap to a lobster and calculates the distance.*

Description

The function finds the closest trap to a lobster and calculates the distance.

Usage

```
distanceToClosestTrap(Lobster, Trap)
```

Arguments

`Lobster`	location of lobster in the arena
`Trap`	location of trap in the arena

Value

Returns distance to closest trap and saves the trap number in case of multiple traps.

`distanceToTrapCalculator` *This function calculates the Euclidean distance between Trap(s) and each lobster. The function is internally called in* `distanceToClosestTrap` *function.*

Description

This function calculates the Euclidean distance between Trap(s) and each lobster. The function is internally called in `distanceToClosestTrap` function.

Usage

```
distanceToTrapCalculator(Lobster, Trap)
```

Arguments

`Lobster`	location of lobster in the grid in x and y coordinates.
`Trap`	location of trap in the grid in x and y coordinates.

Value

Returns the distance to trap.

`GetSimOutput` *This function extracts the results of simulation.*

Description

This function extracts the results of simulation.

Usage

```
GetSimOutput(x, mls = 82.5)
```

Arguments

`x`	is an object generated by `SimulateLobsterMovement` function.
`mls`	is the minimum legal size(mls) in mm. The default is 82.5 mm.

Value

Returns the followings for each replicate: the number of lobsters caught, legal catch weight (bigger than mls), total catch weight and length of time to reach maximum catch.

`initialLobsterGrid` *This function simulates an arena (or grid) with lobsters in it based on the provided density, size and sex distribution.*

Description

This function simulates an arena (or grid) with lobsters in it based on the provided density, size and sex distribution.

Usage

```
initialLobsterGrid(
  nrowgrids,
  ncolgrids,
  unitarea,
  initlambda,
  initD,
  lobsterSizeFile,
  lobsterSexDist
)
```

Arguments

nrowgrids	is a numeric value which defines the number of rows of the arena.
ncolgrids	is a numeric value which defines the number of columns of the arena.
unitarea	is the unit area used for estimating density of lobsters.
initlambda	is the density of lobsters at the beginning of simulation.
initD	is the dispersion index of lobsters on seabed at the beginning of the simulation.
lobsterSizeFile	is a csv file that contains the frequency of lobsters size class.
lobsterSexDist	is a list that contains the sex ratio of lobsters. Possible values are M=male, F=female, MM=mature male, BF=berried female)

Value

Returns x and y coordinates of simulated lobsters at the beginning.

LobsterSizeFreqs *Lobster size frequency data*

Description

The dataset contains frequency of each size bin (from Carapace length of 50 mm to 200 mm)

Usage

```
data(LobsterSizeFreqs)
```

Format

A data frame with 31 rows and 2 variables

Details

- bins (Size groups/bins)
- freq (Frequency)

`randomMove` *The function randomly selects an angle (0:360) and moves the lobster. This function is called when a lobster is outside the area of influence.*

Description

The function randomly selects an angle (0:360) and moves the lobster. This function is called when a lobster is outside the area of influence.

Usage

```
randomMove(Lobster, dStep)
```

Arguments

`Lobster`	location of lobster in x and y coordinates
`dStep`	is how much a lobster moves in each time step

Value

Returns the new coordinates of each lobster

`replicateCoordinates` *This function replicates the coordinates where there are multiple lobsters*

Description

This function replicates the coordinates where there are multiple lobsters

Usage

```
replicateCoordinates(d)
```

Arguments

`d`	is a data frame containing x and y coordinates of lobsters and number of lobsters at each coordinate

Value

Returns a data frame

`rpoisD` *This function generates a Poisson or a negative binomial distribution for lobsters in the arena*

Description

This function generates a Poisson or a negative binomial distribution for lobsters in the arena

Usage

```
rpoisD(n, lambda, D = 1)
```

Arguments

`n`	is the number of lobsters to be generated
`lambda`	is the mean density of lobsters
`D`	is the dispersion index to be used. Default value is 1

Value

A vector of integers that is used as initial distribution of lobsters

`SimulateLobsterMovement` *Function to run the simulation based on defined parameters*

Description

Function to run the simulation based on defined parameters

Usage

```
SimulateLobsterMovement(p)
```

Arguments

`p`	is a list of all input variables

Value

Returns a list

See Also

Examples of the input parameters and more details can be found here: https://github.com/pnickchi/lobstercatch/blob/main/Exa

Examples

```
p = list()
p$nrowgrids = 10
p$ncolgrids = 10
p$ngrids = p$nrowgrids * p$ncolgrids
p$unitarea = 1
p$initlambda = 0.5
p$dStep = 1
p$howClose = 1
p$initD = 1
p$shrinkage = 0.993
p$currentZoI = 15
p$radiusOfInfluence = 15
p$q0 = 0.5
p$qmin = 0
p$Trap = data.frame( x = c(5), y = c(5) )
p$ntraps = nrow(p$Trap)
p$saturationThreshold = 5
p$lengthBased = FALSE
p$lobsterSizeFile =
'https://raw.githubusercontent.com/vpourfaraj/lobsterCatch/main/inst/extdata/LobsterSizeFreqs.csv'
p$lobLengthThreshold = 115
p$trapSaturation = FALSE
p$sexBased = FALSE
p$lobsterSexDist = list(labels = c('M','F','MM','BF'),
                        prob1 = c(0.55,0.35,0.05,0.05),
                        prob2 = c(0.5,0.50,0,0),
                        lobsterMatThreshold = 100)
p$realizations = 2
p$tSteps = 2
Simrun = SimulateLobsterMovement(p)
```

`trapInPath` *This function determines if lobster gets into a trap and is caught.*

Description

This function determines if lobster gets into a trap and is caught.

Usage

```
trapInPath(loc1, loc2, Trap, howClose)
```

Arguments

`loc1`	is the location of lobster at the start of each time step
`loc2`	is the location of lobster at the end of each time step
`Trap`	is the location of trap
`howClose`	The area within which a lobster considered trapped

Value

Returns a vector that contain lobster path and whether its trapped

---

`updateGrid` *This function updates the coordinate of each lobster at each timestep,*

Description

This function updates the coordinate of each lobster at each timestep,

Usage

```
updateGrid(
  Lobster,
  Trap,
  trapCatch,
  lobSize,
  lobSex,
  radiusOfInfluence,
  currentZoI,
  dStep,
  howClose,
  q0,
  qmin,
  saturationThreshold,
  trapSaturation,
  lengthBased,
  lobLengthThreshold,
  sexBased
)
```

Arguments

`Lobster`	is the x & y coordinates of each lobster
`Trap`	is the x & y coordinates of the trap
`trapCatch`	number of trapped lobster
`lobSize`	Size of trapped lobster
`lobSex`	Sex of trapped lobster
`radiusOfInfluence`	is the initial radius of influence
`currentZoI`	is the bait's area of influence at each timestep
`dStep`	is how much a lobster moves in each time step
`howClose`	The area within which a lobster considered trapped
`q0`	is the initial probability of entry into an empty trap

qmin	is the asymptotic minimum probability of entry
saturationThreshold	is the number of lobsters in a trap at which the probability of another lobster entering the trap is zero
trapSaturation	Logical. If TRUE, lobsters behavioral interaction is included during the simulation.
lengthBased	Logical. If TRUE, lobster size is taken into account
lobLengthThreshold	is a size threshold (Carapace Length in mm), if a lobster larger than this threshold caught there will be no more entry to the trap
sexBased	Logical. If TRUE, lobster sex is taken into account and user must provide a list containing sex distribution for the simulated population

Value

a list of new coordinates, number of catch and their sizes

Index