Package ‘nimbleSCR’

October 25, 2021

Type Package

Title Spatial Capture-Recapture (SCR) Methods Using 'nimble'

Version 0.1.3

Maintainer Daniel Turek <danielturek@gmail.com>

Date 2021-10-25

Description Provides utility functions, distributions, and methods for improving Markov chain Monte Carlo (MCMC) sampling efficiency for ecological Spatial Capture-Recapture (SCR) and Open Population Spatial Capture-Recapture (OPSCR) models. The methods provided implement the techniques presented in ``Estimation of Large-Scale Spatial Capture-Recapture Models'' (Turek, et al (2021); Eco-sphere 12(2):e03385. <doi:10.1002/ecs2.3385>).

License GPL-3

Depends R (>= 3.5.0), nimble

Encoding UTF-8

RoxygenNote 7.1.1

Suggests knitr, rmarkdown, testthat (>= 3.0.0), coda, basicMCMCplots

VignetteBuilder knitr

Config/testthat/edition 3

Collate calcWindowSizes.R getWindowIndex.R integrateIntensityLocal_normal.R integrateIntensityLocal_exp.R integrateIntensity_normal.R integrateIntensity_exp.R stratRejectionSampler_normal.R stratRejectionSampler_exp.R dDispersal_exp.R dHabitatMask.R dbernppAC.R dbernppACmovement_normal.R dbernppACmovement_exp.R dbernppDetection_normal.R dbernppLocalACmovement_normal.R dbernppLocalACmovement_exp.R dbernppLocalDetection_normal.R dbinomLocal_normal.R dbinom_sparseLocalSCR.R dbinom_vector.R dnormalizer.R dpoisLocal_normal.R dpoisppAC.R dpoisppDetection_normal.R dpoisppLocalDetection_normal.R getLocalObjects.R getLocalTraps.R getMidPointNodes.R getWindowCoords.R getSparseY.R localTrapCalculations.R makeConstantNimbleFunction.R marginalVoidProbIntegrand.R marginalVoidProbNumIntegration.R scaleCoordsToHabitatGrid.R zzz.R

NeedsCompilation no

Author Richard Bischof [aut],
Daniel Turek [aut, cre],
Cyril Milleret [aut],
Torbjørn Ergon [aut],
Pierre Dupont [aut],
Soumen Dey [aut],
Wei Zhang [aut],
Perry de Valpine [aut]

Repository CRAN

Date/Publication 2021-10-25 15:50:02 UTC

R topics documented:

calcWindowSizes *Window size calculation*

Description

Calculates the sizes of a set of windows based on their lower and upper coordinates of each dimension. Can be applied to detection and habitat windows.

Usage

```
calcWindowSizes(lowerCoords, upperCoords)
```

Arguments

`lowerCoords`	Matrix of lower x- and y-coordinates of all windows. One row for each window.
`upperCoords`	Matrix of upper x- and y-coordinates of all windows. One row for each window.

Value

A vector of window sizes.

Author(s)

Wei Zhang

Examples

```
lowerCoords <- matrix(c(0, 0, 1, 0, 0, 1, 1, 1), nrow = 4, byrow = TRUE)
upperCoords <- matrix(c(1, 1, 3, 1, 1, 4, 3, 4), nrow = 4, byrow = TRUE)
calcWindowSizes(lowerCoords, upperCoords)
```

dbernppAC *Bernoulli point process for the distribution of activity centers*

Description

Density and random generation functions of the Bernoulli point process for the distribution of activity centers.

Usage

```
dbernppAC(
  x,
  lowerCoords,
  upperCoords,
  logIntensities,
  logSumIntensity,
  habitatGrid,
  numGridRows,
  numGridCols,
  log = 0
)

rbernppAC(
  n,
  lowerCoords,
  upperCoords,
  logIntensities,
  logSumIntensity,
  habitatGrid,
  numGridRows,
  numGridCols
)
```

Arguments

`x`	Vector of x- and y-coordinates of a single spatial point (i.e. AC location).
`lowerCoords, upperCoords`	Matrices of lower and upper x- and y-coordinates of all habitat windows. One row for each window. Each window should be of size 1x1 (after rescaling if necessary).
`logIntensities`	Vector of log habitat intensities for all habitat windows.
`logSumIntensity`	Log of the sum of habitat intensities over all windows.
`habitatGrid`	Matrix of habitat window indices. Only needed for `dbernppAC`. Habitat window indices should match the order in `lowerCoords`, `upperCoords`, and `logIntensities`. When the grid has only one row/column, artificial indices have to be provided to inflate `habitatGrid` in order to be able to use the distribution in `nimble` model code.
`numGridRows, numGridCols`	Numbers of rows and columns of the habitat grid.
`log`	Logical argument, specifying whether to return the log-probability of the distribution.
`n`	Integer specifying the number of realisations to generate. Only n = 1 is supported.

Details

The `dbernppAC` distribution is a NIMBLE custom distribution which can be used to model and simulate the activity center location (*x*) of a single individual in continuous space over a set of habitat windows defined by their upper and lower coordinates (*lowerCoords,upperCoords*). The distribution assumes that the activity center follows a Bernoulli point process with intensity = *exp(logIntensities)*.

Value

`dbernppAC` gives the (log) probability density of the observation vector `x`. `rbernppAC` gives coordinates of a randomly generated spatial point.

Author(s)

Wei Zhang

References

W. Zhang, J. D. Chipperfield, J. B. Illian, P. Dupont, C. Milleret, P. de Valpine and R. Bischof. 2020. A hierarchical point process model for spatial capture-recapture data. bioRxiv. DOI 10.1101/2020.10.06.325035

Examples

```
# Use the distribution in R
lowerCoords <- matrix(c(0, 0, 1, 0, 0, 1, 1, 1), nrow = 4, byrow = TRUE)
upperCoords <- matrix(c(1, 1, 2, 1, 1, 2, 2, 2), nrow = 4, byrow = TRUE)
logIntensities <- log(c(1:4))
logSumIntensity <- log(sum(c(1:4)))
habitatGrid <- matrix(c(1:4), nrow = 2, byrow = TRUE)
numGridRows <- nrow(habitatGrid)
numGridCols <- ncol(habitatGrid)
dbernppAC(c(0.5, 1.5), lowerCoords, upperCoords, logIntensities, logSumIntensity,
          habitatGrid, numGridRows, numGridCols, log = TRUE)
```

`dbernppACmovement_exp` *Bernoulli point process for activity center movement (exponential kernel)*

Description

Density and random generation functions of the Bernoulli point process for activity center movement.

Usage

```
dbernppACmovement_exp(
  x,
  lowerCoords,
  upperCoords,
  s,
  lambda,
  baseIntensities,
  habitatGrid,
  numGridRows,
  numGridCols,
  numWindows,
  log = 0
)

rbernppACmovement_exp(
  n,
  lowerCoords,
  upperCoords,
  s,
  lambda,
  baseIntensities,
  habitatGrid,
  numGridRows,
  numGridCols,
  numWindows
)
```

Arguments

`x`	Vector of x- and y-coordinates of a single spatial point (typically AC location at time t+1).
`lowerCoords, upperCoords`	Matrices of lower and upper x- and y-coordinates of all habitat windows. One row for each window. Each window should be of size 1x1 (after rescaling if necessary).
`s`	Vector of x- and y-coordinates of the isotropic multivariate exponential distribution mean (AC location at time t).
`lambda`	Rate parameter of the isotropic multivariate exponential distribution.
`baseIntensities`	Vector of baseline habitat intensities for all habitat windows.
`habitatGrid`	Matrix of habitat window indices. Baseline habitat intensities should match the order in `lowerCoords` and `upperCoords`. When the grid has only one row/column, artificial indices have to be provided to inflate `habitatGrid` in order to be able to use the distribution in `nimble` model code.
`numGridRows, numGridCols`	Numbers of rows and columns of the habitat grid.

`numWindows`	Number of habitat windows. This value (positive integer) can be used to truncate `lowerCoords` and `upperCoords` so that extra rows beyond `numWindows` are ignored.
`log`	Logical argument, specifying whether to return the log-probability of the distribution.
`n`	Integer specifying the number of realisations to generate. Only n = 1 is supported.

Details

The `dbernppACmovement_exp` distribution is a NIMBLE custom distribution which can be used to model and simulate movement of activity centers between consecutive occasions in open population models. The distribution assumes that the new individual activity center location (*x*) follows an isotropic exponential normal centered on the previous activity center (*s*) with rate (*lambda*).

Value

`dbernppACmovement_exp` gives the (log) probability density of the observation vector `x`. `rbernppACmovement_exp` gives coordinates of a randomly generated spatial point.

Author(s)

Wei Zhang and Cyril Milleret

References

W. Zhang, J. D. Chipperfield, J. B. Illian, P. Dupont, C. Milleret, P. de Valpine and R. Bischof. 2020. A hierarchical point process model for spatial capture-recapture data. bioRxiv. DOI 10.1101/2020.10.06.325035

Examples

```
# Use the distribution in R
lowerCoords <- matrix(c(0, 0, 1, 0, 0, 1, 1, 1), nrow = 4, byrow = TRUE)
upperCoords <- matrix(c(1, 1, 2, 1, 1, 2, 2, 2), nrow = 4, byrow = TRUE)
s <- c(1, 1) # Currrent activity center location
lambda <- 0.1
baseIntensities <- c(1:4)
habitatGrid <- matrix(c(1:4), nrow = 2, byrow = TRUE)
numRows <- nrow(habitatGrid)
numCols <- ncol(habitatGrid)
numWindows <- 4
# The log probability density of moving from (1,1) to (1.2, 0.8)
dbernppACmovement_exp(c(1.2, 0.8), lowerCoords, upperCoords, s, lambda, baseIntensities,
                      habitatGrid, numRows, numCols, numWindows, log = TRUE)
```

dbernppACmovement_normal

Bernoulli point process for activity center movement (normal kernel)

Description

Density and random generation functions of the Bernoulli point process for activity center movement.

Usage

```
dbernppACmovement_normal(
  x,
  lowerCoords,
  upperCoords,
  s,
  sd,
  baseIntensities,
  habitatGrid,
  numGridRows,
  numGridCols,
  numWindows,
  log = 0
)

rbernppACmovement_normal(
  n,
  lowerCoords,
  upperCoords,
  s,
  sd,
  baseIntensities,
  habitatGrid,
  numGridRows,
  numGridCols,
  numWindows
)
```

Arguments

`x`	Vector of x- and y-coordinates of a single spatial point (typically AC location at time t+1).
`lowerCoords, upperCoords`	Matrices of lower and upper x- and y-coordinates of all habitat windows. One row for each window. Each window should be of size 1x1 (after rescaling if necessary).

`s`	Vector of x- and y-coordinates of the isotropic multivariate normal distribution mean (AC location at time t).
`sd`	Standard deviation of the isotropic multivariate normal distribution.
`baseIntensities`	Vector of baseline habitat intensities for all habitat windows.
`habitatGrid`	Matrix of habitat window indices. Baseline habitat intensities should match the order in `lowerCoords` and `upperCoords`. When the grid has only one row/column, artificial indices have to be provided to inflate `habitatGrid` in order to be able to use the distribution in `nimble` model code.
`numGridRows, numGridCols`	Numbers of rows and columns of the habitat grid.
`numWindows`	Number of habitat windows. This value (positive integer) can be used to truncate `lowerCoords` and `upperCoords` so that extra rows beyond `numWindows` are ignored.
`log`	Logical argument, specifying whether to return the log-probability of the distribution.
`n`	Integer specifying the number of realisations to generate. Only n = 1 is supported.

Details

The `dbernppACmovement_normal` distribution is a NIMBLE custom distribution which can be used to model and simulate movement of activity centers between consecutive occasions in open population models. The distribution assumes that the new individual activity center location (x) follows an isotropic multivariate normal centered on the previous activity center (s) with standard deviation (sd).

Value

`dbernppACmovement_normal` gives the (log) probability density of the observation vector `x`. `rbernppACmovement_normal` gives coordinates of a randomly generated spatial point.

Author(s)

Wei Zhang and Cyril Milleret

References

W. Zhang, J. D. Chipperfield, J. B. Illian, P. Dupont, C. Milleret, P. de Valpine and R. Bischof. 2020. A hierarchical point process model for spatial capture-recapture data. bioRxiv. DOI 10.1101/2020.10.06.325035

Examples

```
# Use the distribution in R
lowerCoords <- matrix(c(0, 0, 1, 0, 0, 1, 1, 1), nrow = 4, byrow = TRUE)
upperCoords <- matrix(c(1, 1, 2, 1, 1, 2, 2, 2), nrow = 4, byrow = TRUE)
s <- c(1, 1) # Currrent activity center location
sd <- 0.1
```

```
baseIntensities <- c(1:4)
habitatGrid <- matrix(c(1:4), nrow = 2, byrow = TRUE)
numRows <- nrow(habitatGrid)
numCols <- ncol(habitatGrid)
numWindows <- 4
# The log probability density of moving from (1,1) to (1.2, 0.8)
dbernppACmovement_normal(c(1.2, 0.8), lowerCoords, upperCoords, s, sd, baseIntensities,
                         habitatGrid, numRows, numCols, numWindows, log = TRUE)
```

---

dbernppDetection_normal

Bernoulli point process detection model

Description

Density and random generation functions of the Bernoulli point process for detection.

Usage

```
dbernppDetection_normal(
  x,
  lowerCoords,
  upperCoords,
  s,
  sd,
  baseIntensities,
  numWindows,
  indicator,
  log = 0
)

rbernppDetection_normal(
  n,
  lowerCoords,
  upperCoords,
  s,
  sd,
  baseIntensities,
  numWindows,
  indicator
)
```

Arguments

`x`	Vector with three elements representing the x- and y-coordinates and the corresponding id the detection window of a single spatial point (detection location).

`lowerCoords, upperCoords`	Matrices of lower and upper x- and y-coordinates of all detection windows. One row for each window.
`s`	Vector of x- and y-coordinates of the isotropic multivariate normal distribution mean (i.e. the AC location).
`sd`	Standard deviation of the isotropic multivariate normal distribution.
`baseIntensities`	Vector of baseline detection intensities for all detection windows.
`numWindows`	Number of detection windows. This value (positive integer) is used to truncate `lowerCoords` and `upperCoords` so that extra rows beyond `numWindows` are ignored.
`indicator`	Binary variable (0 or 1). `indicator = 0` means the individual is not available for detection and thus the probability of no detection is 1.
`log`	Logical argument, specifying whether to return the log-probability of the distribution.
`n`	Integer specifying the number of realisations to generate. Only n = 1 is supported.

Details

The `dbernppDetection_normal` distribution is a NIMBLE custom distribution which can be used to model and simulate Bernoulli observations (*x*) of a single individual in continuous space over a set of detection windows defined by their upper and lower coordinates (*lowerCoords,upperCoords*). The distribution assumes that an individual's detection probability follows an isotropic multivariate normal centered on the individual's activity center (*s*) with standard deviation (*sd*).

Value

`dbernppDetection_normal` gives the (log) probability density of the observation vector `x`. `rbernppDetection_normal` gives coordinates of a randomly generated spatial point.

Author(s)

Wei Zhang and Cyril Milleret

References

W. Zhang, J. D. Chipperfield, J. B. Illian, P. Dupont, C. Milleret, P. de Valpine and R. Bischof. 2020.
A hierarchical point process model for spatial capture-recapture data. bioRxiv. DOI 10.1101/2020.10.06.325035

Examples

```
coordsHabitatGridCenter <- matrix(c(0.5, 3.5,
                                    1.5, 3.5,
                                    2.5, 3.5,
                                    3.5, 3.5,
                                    0.5, 2.5,
                                    1.5, 2.5,
                                    2.5, 2.5,
                                    3.5, 2.5,
                                    0.5, 1.5,
                                    1.5, 1.5,
                                    2.5, 1.5,
                                    3.5, 1.5,
                                    0.5, 0.5,
                                    1.5, 0.5,
                                    2.5, 0.5,
                                    3.5, 0.5), ncol = 2,byrow = TRUE)
colnames(coordsHabitatGridCenter) <- c("x","y")
# Create observation windows
 lowerCoords <- matrix(c(0, 0, 1, 0, 0, 1, 1, 1), nrow = 4, byrow = TRUE)
 upperCoords <- matrix(c(1, 1, 2, 1, 1, 2, 2, 2), nrow = 4, byrow = TRUE)
 colnames(lowerCoords) <- colnames(upperCoords) <- c("x","y")
# Rescale coordinates
ScaledLowerCoords <- scaleCoordsToHabitatGrid(coordsData =  lowerCoords,
                                      coordsHabitatGridCenter = coordsHabitatGridCenter)
ScaledUpperCoords <- scaleCoordsToHabitatGrid(coordsData =  upperCoords,
                                      coordsHabitatGridCenter = coordsHabitatGridCenter)
ScaledUpperCoords$coordsDataScaled[,2] <- ScaledUpperCoords$coordsDataScaled[,2] + 1.5
ScaledLowerCoords$coordsDataScaled[,2] <- ScaledLowerCoords$coordsDataScaled[,2] - 1.5


s <- c(1, 1)
sd <- 0.1
baseIntensities <- c(1:4)
windowIndex <- 4
numPoints <- 1
numWindows <- 4
indicator <- 1
x <- c(0.5, 2)
windowIndex <- getWindowIndex(curCoords = x,
                              lowerCoords = ScaledLowerCoords$coordsDataScaled,
                              upperCoords =ScaledUpperCoords$coordsDataScaled)
x <- c(x, windowIndex)

dbernppDetection_normal(x, lowerCoords, upperCoords,
                        s, sd, baseIntensities
                        , numWindows,
                        indicator, log = TRUE)
```

dbernppLocalACmovement_exp

Local evaluation of a Bernoulli point process for activity center movement (exponential kernel)

Description

Density and random generation functions of the Bernoulli point process for activity center movement.

Usage

```
dbernppLocalACmovement_exp(
  x,
  lowerCoords,
  upperCoords,
  s,
  lambda,
  baseIntensities,
  habitatGrid,
  habitatGridLocal,
  resizeFactor,
  localHabWindowIndices,
  numLocalHabWindows,
  numGridRows,
  numGridCols,
  numWindows,
  log = 0
)

rbernppLocalACmovement_exp(
  n,
  lowerCoords,
  upperCoords,
  s,
  lambda,
  baseIntensities,
  habitatGrid,
  habitatGridLocal,
  resizeFactor,
  localHabWindowIndices,
  numLocalHabWindows,
  numGridRows,
  numGridCols,
  numWindows
)
```

Arguments

`x`	Vector of x- and y-coordinates of a single spatial point (typically AC location at time t+1).
`lowerCoords, upperCoords`	Matrices of lower and upper x- and y-coordinates of all habitat windows. One row for each window. Each window should be of size 1x1 (after rescaling if necessary).
`s`	Vector of x- and y-coordinates of the isotropic multivariate exponential distribution mean (AC location at time t).
`lambda`	Rate parameter of the isotropic multivariate exponential distribution.

`baseIntensities`	Vector of baseline habitat intensities for all habitat windows.
`habitatGrid`	Matrix of habitat window indices. When the grid has only one row/column, artificial indices have to be provided to inflate `habitatGrid` in order to be able to use the distribution in `nimble` model code.
`habitatGridLocal`	Matrix of rescaled habitat grid cells indices, as returned by the `getLocalObjects` function (object named `habitatGrid`).
`resizeFactor`	Scalar (aggregation factor) for rescaling habitat windows as used in `getLocalObjects`.
`localHabWindowIndices`	Matrix of indices of local habitat windows around each rescaled habitat grid cell, as returned by the getLocalObjects function (object named `localIndices`).
`numLocalHabWindows`	Vector of numbers of local habitat windows around all habitat grid cells, as returned by the getLocalObjects function (object named `numLocalIndices`). The ith number gives the number of local (original) habitat windows for the ith (rescaled) habitat window.
`numGridRows, numGridCols`	Numbers of rows and columns of the `habitatGrid`.
`numWindows`	Number of habitat windows. This value (positive integer) is used to truncate `lowerCoords` and `upperCoords` so that extra rows beyond `numWindows` are ignored.
`log`	Logical argument, specifying whether to return the log-probability of the distribution.
`n`	Integer specifying the number of realisations to generate. Only n = 1 is supported.

Details

The `dbernppLocalACmovement_exp` distribution is a NIMBLE custom distribution which can be used to model and simulate movement of activity centers between consecutive occasions in open population models. The distribution assumes that the new individual activity center location (*x*) follows an isotropic exponential normal centered on the previous activity center (*s*) with rate (*lambda*). The local evaluation approach is implemented.

Value

The (log) probability density of the observation vector `x`.

Author(s)

Wei Zhang and Cyril Milleret

References

W. Zhang, J. D. Chipperfield, J. B. Illian, P. Dupont, C. Milleret, P. de Valpine and R. Bischof. 2020. A hierarchical point process model for spatial capture-recapture data. bioRxiv. DOI 10.1101/2020.10.06.325035

C. Milleret, P. Dupont, C. Bonenfant, H. Broseth, O. Flagstad, C. Sutherland and R. Bischof. 2019. A local evaluation of the individual state-space to scale up Bayesian spatial capture-recapture. Ecology and Evolution 9:352-363

Examples

```
# Creat habitat grid
habitatGrid <- matrix(c(1:(4^2)), nrow = 4, ncol=4, byrow = TRUE)
coordsHabitatGridCenter <- matrix(c(0.5, 3.5,
                                    1.5, 3.5,
                                    2.5, 3.5,
                                    3.5, 3.5,
                                    0.5, 2.5,
                                    1.5, 2.5,
                                    2.5, 2.5,
                                    3.5, 2.5,
                                    0.5, 1.5,
                                    1.5, 1.5,
                                    2.5, 1.5,
                                    3.5, 1.5,
                                    0.5, 0.5,
                                    1.5, 0.5,
                                    2.5, 0.5,
                                    3.5, 0.5), ncol = 2,byrow = TRUE)
colnames(coordsHabitatGridCenter) <- c("x","y")
# Create habitat windows
lowerCoords <- coordsHabitatGridCenter-0.5
upperCoords <- coordsHabitatGridCenter+0.5
colnames(lowerCoords) <- colnames(upperCoords) <- c("x","y")
# Plot check
plot(lowerCoords[,"y"]~lowerCoords[,"x"],pch=16, xlim=c(0,4), ylim=c(0,4),col="red")
points(upperCoords[,"y"]~upperCoords[,"x"],col="red",pch=16)
points(coordsHabitatGridCenter[,"y"]~coordsHabitatGridCenter[,"x"],pch=16)

# Rescale coordinates
ScaledLowerCoords <- scaleCoordsToHabitatGrid(coordsData =  lowerCoords,
                                      coordsHabitatGridCenter = coordsHabitatGridCenter)
ScaledUpperCoords <- scaleCoordsToHabitatGrid(coordsData =  upperCoords,
                                      coordsHabitatGridCenter = coordsHabitatGridCenter)
ScaledUpperCoords$coordsDataScaled[,2] <- ScaledUpperCoords$coordsDataScaled[,2] + 1
ScaledLowerCoords$coordsDataScaled[,2] <- ScaledLowerCoords$coordsDataScaled[,2] - 1
habitatMask <- matrix(1, nrow = 4, ncol=4, byrow = TRUE)
# Create local objects
HabWindowsLocal <- getLocalObjects(habitatMask = habitatMask,
                                   coords = coordsHabitatGridCenter,
                                   dmax=4,
                                   resizeFactor = 1,
                                   plot.check = TRUE
)

s <- c(1, 1) # Currrent activity center location
lambda <- 0.1
```

```
numWindows <- nrow(coordsHabitatGridCenter)
baseIntensities <- rep(1,numWindows)
numRows <- nrow(habitatGrid)
numCols <- ncol(habitatGrid)

# The log probability density of moving from (1,1) to (1.2, 0.8)
dbernppLocalACmovement_exp(x = c(1.2, 0.8), lowerCoords, upperCoords, s,
                              lambda, baseIntensities, habitatGrid,
                              HabWindowsLocal$habitatGrid, HabWindowsLocal$resizeFactor,
                          HabWindowsLocal$localIndices, HabWindowsLocal$numLocalIndices,
                              numRows, numCols, numWindows, log = TRUE)
```

dbernppLocalACmovement_normal

Local evaluation of a Bernoulli point process for activity center movement (normal kernel)

Description

Density and random generation functions of the Bernoulli point process for activity center movement.

Usage

```
dbernppLocalACmovement_normal(
  x,
  lowerCoords,
  upperCoords,
  s,
  sd,
  baseIntensities,
  habitatGrid,
  habitatGridLocal,
  resizeFactor,
  localHabWindowIndices,
  numLocalHabWindows,
  numGridRows,
  numGridCols,
  numWindows,
  log = 0
)

rbernppLocalACmovement_normal(
  n,
  lowerCoords,
  upperCoords,
  s,
  sd,
  baseIntensities,
  habitatGrid,
  habitatGridLocal,
  resizeFactor,
  localHabWindowIndices,
  numLocalHabWindows,
  numGridRows,
  numGridCols,
  numWindows
)
```

Arguments

`x`	Vector of x- and y-coordinates of a single spatial point (typically AC location at time t+1).
`lowerCoords, upperCoords`	Matrices of lower and upper x- and y-coordinates of all habitat windows. One row for each window. Each window should be of size 1x1 (after rescaling if necessary).
`s`	Vector of x- and y-coordinates of the isotropic multivariate normal distribution mean (AC location at time t).
`sd`	Standard deviation of the isotropic multivariate normal distribution.
`baseIntensities`	Vector of baseline habitat intensities for all habitat windows.
`habitatGrid`	Matrix of habitat window indices. When the grid has only one row/column, artificial indices have to be provided to inflate `habitatGrid` in order to be able to use the distribution in `nimble` model code.
`habitatGridLocal`	Matrix of rescaled habitat grid cells indices, as returned by the `getLocalObjects` function (object named `habitatGrid`).
`resizeFactor`	Scalar (aggregation factor) for rescaling habitat windows as used in `getLocalObjects`.
`localHabWindowIndices`	Matrix of indices of local habitat windows around each rescaled habitat grid cell, as returned by the getLocalObjects function (object named `localIndices`).
`numLocalHabWindows`	Vector of numbers of local habitat windows around all habitat grid cells, as returned by the getLocalObjects function (object named `numLocalIndices`). The ith number gives the number of local (original) habitat windows for the ith (rescaled) habitat window.
`numGridRows, numGridCols`	Numbers of rows and columns of the `habitatGrid`.
`numWindows`	Number of habitat windows. This value (positive integer) is used to truncate `lowerCoords` and `upperCoords` so that extra rows beyond `numWindows` are ignored.

`log`	Logical argument, specifying whether to return the log-probability of the distribution.
`n`	Integer specifying the number of realisations to generate. Only n = 1 is supported.

Details

The `dbernppLocalACmovement_normal` distribution is a NIMBLE custom distribution which can be used to model and simulate movement of activity centers between consecutive occasions in open population models. The distribution assumes that the new individual activity center location (x) follows an isotropic multivariate normal centered on the previous activity center (s) with standard deviation (sd). The local evaluation technique is implemented.

Value

The (log) probability density of the observation vector `x`.

Author(s)

Wei Zhang and Cyril Milleret

References

W. Zhang, J. D. Chipperfield, J. B. Illian, P. Dupont, C. Milleret, P. de Valpine and R. Bischof. 2020. A hierarchical point process model for spatial capture-recapture data. bioRxiv. DOI 10.1101/2020.10.06.325035

C. Milleret, P. Dupont, C. Bonenfant, H. Brøseth, Ø. Flagstad, C. Sutherland and R. Bischof. 2019. A local evaluation of the individual state-space to scale up Bayesian spatial capture-recapture. Ecology and Evolution 9:352-363

Examples

```
# Creat habitat grid
habitatGrid <- matrix(c(1:(4^2)), nrow = 4, ncol=4, byrow = TRUE)
coordsHabitatGridCenter <- matrix(c(0.5, 3.5,
                                    1.5, 3.5,
                                    2.5, 3.5,
                                    3.5, 3.5,
                                    0.5, 2.5,
                                    1.5, 2.5,
                                    2.5, 2.5,
                                    3.5, 2.5,
                                    0.5, 1.5,
                                    1.5, 1.5,
                                    2.5, 1.5,
                                    3.5, 1.5,
                                    0.5, 0.5,
                                    1.5, 0.5,
                                    2.5, 0.5,
                                    3.5, 0.5), ncol = 2,byrow = TRUE)
colnames(coordsHabitatGridCenter) <- c("x","y")
```

```
# Create habitat windows
lowerCoords <- coordsHabitatGridCenter-0.5
upperCoords <- coordsHabitatGridCenter+0.5
colnames(lowerCoords) <- colnames(upperCoords) <- c("x","y")
# Plot check
plot(lowerCoords[,"y"]~lowerCoords[,"x"],pch=16, xlim=c(0,4), ylim=c(0,4),col="red")
points(upperCoords[,"y"]~upperCoords[,"x"],col="red",pch=16)
points(coordsHabitatGridCenter[,"y"]~coordsHabitatGridCenter[,"x"],pch=16)

# Rescale coordinates
ScaledLowerCoords <- scaleCoordsToHabitatGrid(coordsData =  lowerCoords,
                                   coordsHabitatGridCenter = coordsHabitatGridCenter)
ScaledUpperCoords <- scaleCoordsToHabitatGrid(coordsData =  upperCoords,
                                   coordsHabitatGridCenter = coordsHabitatGridCenter)
ScaledUpperCoords$coordsDataScaled[,2] <- ScaledUpperCoords$coordsDataScaled[,2] + 1
ScaledLowerCoords$coordsDataScaled[,2] <- ScaledLowerCoords$coordsDataScaled[,2] - 1
habitatMask <- matrix(1, nrow = 4, ncol=4, byrow = TRUE)
# Create local objects
HabWindowsLocal <- getLocalObjects(habitatMask = habitatMask,
                                   coords = coordsHabitatGridCenter,
                                   dmax=4,
                                   resizeFactor = 1,
                                   plot.check = TRUE
)

s <- c(1, 1) # Currrent activity center location
sd <- 0.1
numWindows <- nrow(coordsHabitatGridCenter)
baseIntensities <- rep(1,numWindows)
numRows <- nrow(habitatGrid)
numCols <- ncol(habitatGrid)

# The log probability density of moving from (1,1) to (1.2, 0.8)
dbernppLocalACmovement_normal(x = c(1.2, 0.8), lowerCoords, upperCoords, s,
                              sd, baseIntensities, habitatGrid,
                              HabWindowsLocal$habitatGrid, HabWindowsLocal$resizeFactor,
                          HabWindowsLocal$localIndices, HabWindowsLocal$numLocalIndices,
                              numRows, numCols, numWindows, log = TRUE)
```

`dbernppLocalDetection_normal`
Local evaluation for a Bernoulli point process detection model

---

Description

Density and random generation functions of the Bernoulli point process for detection.

Usage

```
dbernppLocalDetection_normal(
  x,
  lowerCoords,
  upperCoords,
  s,
  sd,
  baseIntensities,
  habitatGridLocal,
  resizeFactor,
  localObsWindowIndices,
  numLocalObsWindows,
  numWindows,
  indicator,
  log = 0
)

rbernppLocalDetection_normal(
  n,
  lowerCoords,
  upperCoords,
  s,
  sd,
  baseIntensities,
  habitatGridLocal,
  resizeFactor,
  localObsWindowIndices,
  numLocalObsWindows,
  numWindows,
  indicator
)
```

Arguments

`x`	Vector with three elements representing the x- and y-coordinates (x[1:2]), and the corresponding id the detection window (x[3]) of a single spatial point (detection location).
`lowerCoords, upperCoords`	Matrices of lower and upper x- and y-coordinates of all detection windows. One row for each window.
`s`	Vector of x- and y-coordinates of the isotropic multivariate normal distribution mean (i.e. the AC location).
`sd`	Standard deviation of the isotropic multivariate normal distribution.
`baseIntensities`	Vector of baseline detection intensities for all detection windows.
`habitatGridLocal`	Matrix of rescaled habitat grid cells indices, as returned by the `getLocalObjects` function (object named `habitatGrid`).

`resizeFactor`	Scalar (aggregation factor) for rescaling habitat windows as used in `getLocalObjects`.
`localObsWindowIndices`	Matrix of indices of local observation windows around each rescaled habitat grid cell, as returned by the getLocalObjects function (object named `localIndices`).
`numLocalObsWindows`	Vector of numbers of local observation windows around all habitat grid cells, as returned by the getLocalObjects function (object named `numLocalIndices`). The ith number gives the number of local (original) observation windows for the ith (rescaled) habitat window.
`numWindows`	Number of detection windows. This value (positive integer) is used to truncate `lowerCoords` and `upperCoords` so that extra rows beyond `numWindows` are ignored.
`indicator`	Binary variable (0 or 1). `indicator = 0` means the individual is not available for detection and thus the probability of no detection is 1.
`log`	Logical argument, specifying whether to return the log-probability of the distribution.
`n`	Integer specifying the number of realizations to generate. Only n = 1 is supported.

Details

The `dbernppDetection_normal` distribution is a NIMBLE custom distribution which can be used to model and simulate Bernoulli observations (*x*) of a single individual in continuous space over a set of detection windows defined by their upper and lower coordinates (*lowerCoords,upperCoords*). The distribution assumes that an individual's detection probability follows an isotropic multivariate normal centered on the individual's activity center (*s*) with standard deviation (*sd*). The local evaluation approach is implemented.

Value

The (log) probability density of the observation vector `x`.

Author(s)

Wei Zhang and Cyril Milleret

References

W. Zhang, J. D. Chipperfield, J. B. Illian, P. Dupont, C. Milleret, P. de Valpine and R. Bischof. 2020. A hierarchical point process model for spatial capture-recapture data. bioRxiv. DOI 10.1101/2020.10.06.325035

C. Milleret, P. Dupont, C. Bonenfant, H. Brøseth, Ø. Flagstad, C. Sutherland and R. Bischof. 2019. A local evaluation of the individual state-space to scale up Bayesian spatial capture-recapture. Ecology and Evolution 9:352-363

Examples

```
# Create habitat grid
coordsHabitatGridCenter <- matrix(c(0.5, 3.5,
                                    1.5, 3.5,
                                    2.5, 3.5,
                                    3.5, 3.5,
                                    0.5, 2.5,
                                    1.5, 2.5,
                                    2.5, 2.5,
                                    3.5, 2.5,
                                    0.5, 1.5,
                                    1.5, 1.5,
                                    2.5, 1.5,
                                    3.5, 1.5,
                                    0.5, 0.5,
                                    1.5, 0.5,
                                    2.5, 0.5,
                                    3.5, 0.5), ncol = 2,byrow = TRUE)
colnames(coordsHabitatGridCenter) <- c("x","y")
# Create observation windows
lowerCoords <- matrix(c(1, 1, 2, 1, 1, 2, 2, 2), nrow = 4, byrow = TRUE)
upperCoords <- matrix(c(2, 2, 3, 2, 2, 3, 3, 3), nrow = 4, byrow = TRUE)
colnames(lowerCoords) <- colnames(upperCoords) <- c("x","y")
# Plot check
plot(coordsHabitatGridCenter[,"y"]~coordsHabitatGridCenter[,"x"],pch=16)
points(lowerCoords[,"y"]~lowerCoords[,"x"],col="red",pch=16)
points(upperCoords[,"y"]~upperCoords[,"x"],col="red",pch=16)
#'
s <- c(1, 1)
sd <- 0.1
baseIntensities <- c(1:4)
windowIndex <- 4
numPoints <- 1
numWindows <- 4
indicator <- 1

# Rescale coordinates
ScaledLowerCoords <- scaleCoordsToHabitatGrid(coordsData =  lowerCoords,
                                      coordsHabitatGridCenter = coordsHabitatGridCenter)
ScaledUpperCoords <- scaleCoordsToHabitatGrid(coordsData =  upperCoords,
                                      coordsHabitatGridCenter = coordsHabitatGridCenter)
ScaledUpperCoords$coordsDataScaled[,2] <- ScaledUpperCoords$coordsDataScaled[,2] + 1.5
ScaledLowerCoords$coordsDataScaled[,2] <- ScaledLowerCoords$coordsDataScaled[,2] - 1.5
habitatMask <- matrix(1, nrow = 4, ncol=4, byrow = TRUE)
# Create local objects
ObsWindowsLocal <- getLocalObjects(habitatMask = habitatMask,
                                   coords = ScaledLowerCoords$coordsDataScaled,
                                   dmax=3,
                                   resizeFactor = 1,
                                   plot.check = TRUE
)
```

```
x <- c(1.1, 1.2)
windowIndex <- getWindowIndex(curCoords = x,
                              lowerCoords = ScaledLowerCoords$coordsDataScaled,
                              upperCoords =ScaledUpperCoords$coordsDataScaled)
x <- c(x, windowIndex)
dbernppLocalDetection_normal(x, ScaledLowerCoords$coordsDataScaled,
                             ScaledUpperCoords$coordsDataScaled,
                             s, sd, baseIntensities,
                             ObsWindowsLocal$habitatGrid, ObsWindowsLocal$resizeFactor,
                           ObsWindowsLocal$localIndices,ObsWindowsLocal$numLocalIndices,
                             numWindows, indicator, log = TRUE)
```

`dbinomLocal_normal` *Local evaluation of a binomial SCR detection process*

Description

The `dbinomLocal_normal` distribution is a NIMBLE custom distribution which can be used to model and simulate binomial observations (*x*) of a single individual over a set of detectors defined by their coordinates (*trapCoords*). The distribution assumes that an individual's detection probability at any detector follows a half-normal function of the distance between the individual's activity center (*s*) and the detector location.

Usage

```
dbinomLocal_normal(
  x,
  detNums = -999,
  detIndices,
  size,
  p0 = -999,
  p0Traps,
  sigma,
  s,
  trapCoords,
  localTrapsIndices,
  localTrapsNum,
  resizeFactor = 1,
  habitatGrid,
  indicator,
  lengthYCombined = 0,
  log = 0
)

rbinomLocal_normal(
  n = 1,
  detNums = -999,
  detIndices,
  size,
  p0 = -999,
  p0Traps,
  sigma,
  s,
  trapCoords,
  localTrapsIndices,
  localTrapsNum,
  resizeFactor = 1,
  habitatGrid,
  indicator,
  lengthYCombined = 0
)
```

Arguments

`x`	Vector of individual detection frequencies. This argument can be provided in two formats: (i) with the *y* object as returned by the `getSparseY` function; (ii) with the *yCombined* object as returned by `getSparseY`. Note that when the random generation functionality is used (`rbinomLocal_normal`), only the *yCombined* format can be used. The *yCombined* object combines *detNums*, *x*, and *detIndices* (in that order). When such consolidated representation of the detection data *x* is used, *detIndices* and *detNums* arguments shouldn't be specified.
`detNums`	Number of detections recorded in *x*, as returned by the *detNums* object from the `getSparseY` function. This argument should not be specified when the *yCombined* object (returned by `getSparseY`) is provided as *x*, and when detection data are simulated.
`detIndices`	Vector of indices of traps where the detections in x were recorded, as returned by the *detIndices* object from the `getSparseY` function. This argument should not be specified when *x* is provided as the *yCombined* object (returned by `getSparseY`) and when detection data are simulated.
`size`	Vector of the number of trials (zero or more) for each trap (*trapCoords*).
`p0`	Baseline detection probability used in the half-normal detection function.
`p0Traps`	Vector of baseline detection probabilities for each trap used in the half-normal detection function. When *p0Traps* is used, *p0* should not be provided.
`sigma`	Scale parameter of the half-normal detection function.
`s`	Individual activity center x- and y-coordinates.
`trapCoords`	Matrix of x- and y-coordinates of all traps.
`localTrapsIndices`	Matrix of indices of local traps around each habitat grid cell, as returned by the `getLocalObjects` function.
`localTrapsNum`	Vector of numbers of local traps around all habitat grid cells, as returned by the `getLocalObjects` function.
`resizeFactor`	Aggregation factor used in the `getLocalObjects` function to reduce the number of habitat grid cells to retrieve local traps for.

`habitatGrid`	Matrix of habitat grid cells indices, as returned by the `getLocalObjects` function.
`indicator`	Logical argument specifying whether the individual is available for detection.
`lengthYCombined`	The length of the x argument when the (*yCombined*) format of the detection data is provided (as returned by the *lengthYCombined* object from `getSparseY`).
`log`	Logical argument, specifying whether to return the log-probability of the distribution.
`n`	Integer specifying the number of realizations to generate. Only n = 1 is supported.

Details

The `dbinomLocal_normal` distribution incorporates three features to increase computation efficiency (see Turek et al., 2021 <doi.org/10.1002/ecs2.3385> for more details):

1. A local evaluation of the detection probability calculation (see Milleret et al., 2019 <doi:10.1002/ece3.4751> for more details)
2. A sparse matrix representation (*x*, *detIndices* and *detNums*) of the observation data to reduce the size of objects to be processed.
3. An indicator (*indicator*) to shortcut calculations for individuals unavailable for detection.

The `dbinomLocal_normal` distribution requires x- and y- detector coordinates (*trapCoords*) to be scaled to the habitat grid (*habitatGrid*) using the (`scaleCoordsToHabitatGrid` function.)

When the aim is to simulate detection data:

1. *x* should be provided using the *yCombined* object as returned by `getSparseY`,
2. arguments *detIndices* and *detNums* should not be provided,
3. argument *lengthYCombined* should be provided using the *lengthYCombined* object as returned by `getSparseY`.

Value

The log-likelihood value associated with the vector of detections, given the location of the activity center (s), and the half-normal detection function : $p = p0 * exp(-d^2/\sigma^2)$.

Author(s)

Cyril Milleret, Soumen Dey

Examples

```
# I. DATA SET UP
coordsHabitatGridCenter <- matrix(c(0.5, 3.5,
                                    1.5, 3.5,
                                    2.5, 3.5,
                                    3.5, 3.5,
                                    0.5, 2.5,
                                    1.5, 2.5,
                                    2.5, 2.5,
                                    3.5, 2.5,
                                    0.5, 1.5,
                                    1.5, 1.5,
                                    2.5, 1.5,
                                    3.5, 1.5,
                                    0.5, 0.5,
                                    1.5, 0.5,
                                    2.5, 0.5,
                                    3.5, 0.5), ncol=2,byrow = TRUE)
colnames(coordsHabitatGridCenter) <- c("x","y")
# CREATE OBSERVATION WINDOWS
trapCoords <- matrix(c(1.5, 1.5, 2.5, 1.5, 1.5, 2.5, 2.5, 2.5), nrow = 4, byrow = TRUE)
colnames(trapCoords) <- c("x","y")
# PLOT CHECK
plot(coordsHabitatGridCenter[,"y"]~coordsHabitatGridCenter[,"x"],pch=16)
points(trapCoords[,"y"]~trapCoords[,"x"],col="red",pch=16)

# PARAMETERS
p0 <- 0.2
sigma <- 2
indicator <- 1
# WE CONSIDER 2 INDIVIDUALS
y <- matrix(c(0, 1, 1, 0,
              0, 1, 0, 1),ncol=4,nrow=2)
s <- matrix(c(0.5, 1,
              1.6, 2.3),ncol=2,nrow=2)

# RESCALE COORDINATES
ScaledtrapCoords <- scaleCoordsToHabitatGrid(coordsData =  trapCoords,
                                      coordsHabitatGridCenter = coordsHabitatGridCenter)
ScaledtrapCoords<- ScaledtrapCoords$coordsDataScaled
habitatMask <- matrix(1, nrow = 4, ncol=4, byrow = TRUE)


# CREATE LOCAL OBJECTS
TrapLocal <- getLocalObjects(habitatMask = habitatMask,
                                   coords = ScaledtrapCoords,
                                   dmax=2.5,
                                   resizeFactor = 1,
                                   plot.check = TRUE
)

# GET SPARSE MATRIX
SparseY <- getSparseY(y)

# II. USING THE DENSITY FUNCTION
 # WE TAKE THE FIRST INDIVIDUAL
i=1
  # OPTION 1: USING THE RANDOM GENERATION FUNCTIONNALITY
dbinomLocal_normal(x=SparseY$y[i,,1],
                   detNums=SparseY$detNums[i],
                   detIndices=SparseY$detIndices[i,,1],
                   size=rep(1,4),
                   p0 = p0,
                   sigma= sigma,
                   s=s[i,1:2],
                   trapCoords=ScaledtrapCoords,
                   localTrapsIndices=TrapLocal$localIndices,
                   localTrapsNum=TrapLocal$numLocalIndices,
                   resizeFactor=TrapLocal$resizeFactor,
                   habitatGrid=TrapLocal$habitatGrid,
                   indicator=indicator)

  # OPTION 2: USING RANDOM GENERATION FUNCTIONNALITY
  # WE DO NOT PROVIDE THE detNums AND detIndices ARGUMENTS
dbinomLocal_normal(x=SparseY$yCombined[i,,1],
                   size=rep(1,4),
                   p0 = p0,
                   sigma= sigma,
                   s=s[i,1:2],
                   trapCoords=ScaledtrapCoords,
                   localTrapsIndices=TrapLocal$localIndices,
                   localTrapsNum=TrapLocal$numLocalIndices,
                   resizeFactor=TrapLocal$resizeFactor,
                   habitatGrid=TrapLocal$habitatGrid,
                   indicator=indicator,
                   lengthYCombined = SparseY$lengthYCombined)

# III. USING THE RANDOM GENERATION FUNCTION
rbinomLocal_normal(n=1,
                   size=rep(1,4),
                   p0 = p0,
                   sigma= sigma,
                   s=s[i,1:2],
                   trapCoords=ScaledtrapCoords,
                   localTrapsIndices=TrapLocal$localIndices,
                   localTrapsNum=TrapLocal$numLocalIndices,
                   resizeFactor=TrapLocal$resizeFactor,
                   habitatGrid=TrapLocal$habitatGrid,
                   indicator=indicator,
                   lengthYCombined = SparseY$lengthYCombined)
```

`dbinom_sparseLocalSCR` *Local evaluation of a binomial SCR observation process. This function is deprecated, use* `dbinomLocal_normal` *instead.*

Description

The `dbinom_sparseLocalSCR` distribution is a NIMBLE custom distribution which can be used to model the binomial observations (x) of a single individual over a set of detectors defined by their coordinates (trapCoords). The distribution assumes that the detection probability at any detector follows a half-normal function of the distance between the individual's activity center (s) and the detector location.

Usage

```
dbinom_sparseLocalSCR(
  x,
  detNums,
  detIndices,
  size,
  p0,
  sigma,
  s,
  trapCoords,
  localTrapsIndices,
  localTrapsNum,
  resizeFactor = 1,
  habitatGrid,
  indicator = 1,
  log = 0
)

rbinom_sparseLocalSCR(
  n = 1,
  detNums,
  detIndices,
  size,
  p0,
  sigma,
  s,
  trapCoords,
  localTrapsIndices,
  localTrapsNum,
  resizeFactor = 1,
  habitatGrid,
  indicator = 1
)
```

Arguments

`x`	Vector of individual detection frequencies, as returned by the `getSparseY` function (padded with -1's to maintain the square structure of the observation data).
`detNums`	Number of detections recorded in x, as returned by the `getSparseY` function.
`detIndices`	Vector of the detector indices where the detections in x were recorded, as returned by the `getSparseY` function.
`size`	Vector of the number of trials (zero or more) for each trap (trapCoords).
`p0`	Baseline detection probability used in the half-normal detection function.

`sigma`	Scale parameter of the half-normal detection function.
`s`	Bivariate individual activity center coordinates.
`trapCoords`	Matrix of x- and y-coordinates of all traps.
`localTrapsIndices`	Matrix of indices of local traps around each habitat grid cell, as returned by the `getLocalTraps` function.
`localTrapsNum`	Vector of numbers of local traps around all habitat grid cells, as returned by the getLocalTraps function.
`resizeFactor`	Aggregation factor used in the `getLocalTraps` function to reduce the number of habitat grid cells to retrieve local traps for.
`habitatGrid`	Matrix of habitat grid cells indices.
`indicator`	Logical argument, specifying whether the individual is available for detection.
`log`	Logical argument, specifying whether to return the log-probability of the distribution.
`n`	Integer specifying the number of realisations to generate. Only n = 1 is supported.

Details

The dbinom_sparseLocalSCR distribution incorporates three features to increase computation efficiency:

1. A local evaluation of the detection probability calculation (see Milleret et al. (2019) <doi:10.1002/ece3.4751> for more details).
2. It uses a sparse matrix representation (x, detIndices, detNums) of the observation data to reduce the size of objects to be processed.
3. It uses an indicator (indicator) to shortcut calculations for individuals unavailable for detection.

Value

The log-likelihood value associated with the vector of detections, given the location of the activity center (s), and the half-normal detection function : $p = p0 * exp(-d^2/\sigma^2)$.

Author(s)

Cyril Milleret

Examples

```
## define model code
code <- nimbleCode({
    psi ~ dunif(0,1)
    p0 ~ dunif(0,1)
    sigma ~ dunif(0,100)
    N <- sum(z[1:M])
    for(i in 1:M) {
        s[i, 1] ~ dunif(0, 100)
        s[i, 2] ~ dunif(0, 100)
        z[i] ~ dbern(psi)
        y[i,1:maxDetNum] ~ dbinom_sparseLocalSCR(detNums,
                                                 detIndices,
                                                 size,
                                                 p0,
                                                 sigma,
                                                 s[i,1:2],
                                                 trapCoords,
                                                 localTrapsIndices,
                                                 localTrapsNum,
                                                 resizeFactor,
                                                 habitatGrid,
                                                 z[i])
    }
})

## create NIMBLE model object
## Rmodel <- nimbleModel(code, ...)

## use model object for MCMC, etc.
```

dbinom_vector *Vectorized binomial distribution*

Description

The dbinom_vector distribution is a vectorized version of the binomial distribution. It can be used to model a vector of binomial realizations. NB: using the vectorized version is beneficial only when the entire joint likelihood of the vector of binomial realizations (x) is calculated simultaneously.

Usage

```
dbinom_vector(x, size, prob, log = 0)

rbinom_vector(n = 1, size, prob)
```

Arguments

x	Vector of quantiles.
size	Vector of number of trials (zero or more).
prob	Vector of success probabilities on each trial
log	Logical argument, specifying whether to return the log-probability of the distribution.
n	Number of observations. Only n = 1 is supported.

Value

The log-likelihood value associated with the vector of binomial observations.

Author(s)

Pierre Dupont

Examples

```
## define vectorized model code
code <- nimbleCode({
    p ~ dunif(0,1)
    p_vector[1:J] <- p
    y[1:J] ~ dbinom_vector(size = trials[1:J],
                           prob = p_vector[1:J])
})

## simulate binomial data
J <- 1000
trials <- sample(x = 10, size = J, replace = TRUE)
y <- rbinom_vector(J, size = trials, prob = 0.21)

constants <- list(J = J, trials = trials)

data <- list(y = y)

inits <- list(p = 0.5)

## create NIMBLE model object
Rmodel <- nimbleModel(code, constants, data, inits)

## use model object for MCMC, etc.
```

`dDispersal_exp` *Bivariate exponential dispersal distribution for activity centers*

Description

The dDispersal_exp distribution is a bivariate distribution which can be used to model the latent bivariate activity centers (ACs) of individuals in a population. This distribution models the situation when individual AC dispersal is uniform in direction (that is, dispersal occurs in a direction theta, where theta is uniformly distributed on [-pi, pi]), and with an exponential distribution for the radial dispersal distance.

Usage

```
dDispersal_exp(x, s, rate, log)

rDispersal_exp(n, s, rate)
```

Arguments

`x`	Bivariate activity center coordinates (at time t+1).
`s`	Current location of the bivariate activity center (at time t).
`rate`	Rate parameter of the exponential distribution for dispersal distance.
`log`	Logical argument, specifying whether to return the log-probability of the distribution.
`n`	Integer specifying the number of realisations to generate. Only n = 1 is supported.

Details

The dDispersal_exp distribution models the location of an AC at time (t+1), conditional on the previous AC location at time (t) and the rate parameter (rate) of the exponential distribution for dispersal distance.

Value

The log-probability value associated with the bivariate activity center location x, given the current activity center s, and the rate parameter of the exponential dispersal distance distribution.

Author(s)

Daniel Turek

Examples

```
## define model code
code <- nimbleCode({
    lambda ~ dgamma(0.001, 0.001)
    for(i in 1:N) {
        AC[i, 1, 1] ~ dunif(0, 100)
        AC[i, 2, 1] ~ dunif(0, 100)
        for(t in 2:T) {
            AC[i, 1:2, t+1] ~ dDispersal_exp(s = AC[i, 1:2, t], rate = lambda)
        }
    }
})

constants <- list(N = 10, T = 6)

## create NIMBLE model object
Rmodel <- nimbleModel(code, constants)

## use model object for MCMC, etc.
```

dHabitatMask — *Ones trick distribution for irregular habitat shapes*

Description

The dHabitatMask distribution checks and ensures that the proposed activity center location (s) falls within the suitable habitat (defined in the binary matrix habitatMask).

Usage

```
dHabitatMask(x, s, xmax, xmin, ymax, ymin, habitatMask, log = 0)

rHabitatMask(n, s, xmax, xmin, ymax, ymin, habitatMask)
```

Arguments

`x`	Ones trick data.
`s`	Bivariate activity center coordinates.
`xmax`	Maximum of trap location x-coordinates.
`xmin`	Minimum of trap location x-coordinates.
`ymax`	Maximum of trap location y-coordinates.
`ymin`	Minimum of trap location y-coordinates.
`habitatMask`	A binary matrix object indicating which cells are considered as suitable habitat.
`log`	Logical argument, specifying whether to return the log-probability of the distribution.
`n`	Integer specifying the number of realisations to generate. Only n = 1 is supported.

Details

The rHabitatMask function returns the value of the habitat mask cell (0 or 1) where the proposed activity center falls. See also M. Meredith: SECR in BUGS/JAGS with patchy habitat.

Value

The log-likelihood value associated with the bivariate activity center location s being in the suitable habitat (i.e. 0 if it falls within the habitat mask and -Inf otherwise).

Author(s)

Daniel Turek

Examples

```
## define model code
code <- nimbleCode({
    for(i in 1:N) {
        s[i, 1] ~ dunif(0, 100)
        s[i, 2] ~ dunif(0, 100)
        OK[i] ~ dHabitatMask( s = s[i,1:2],
                              xmax = 100,
                              xmin = 0,
                              ymax = 100,
                              ymin = 0,
                              habitatMask = habitatMask[1:100,1:100])
    }
})

N <- 20

habitatMask <- matrix(rbinom(10000,1,0.75), nrow = 100)

constants <- list(N = N, habitatMask = habitatMask)

data <- list(OK = rep(1, N))

inits <- list(s = array(runif(2*N, 0, 100), c(N,2)))

## create NIMBLE model object
Rmodel <- nimbleModel(code, constants, data, inits)

## use model object for MCMC, etc.
```

`dnormalizer` *Normalizing constant generator*

Description

A normalizer used for normalizing `nimble` distributions. It is particularly useful for fitting `dpoisppDetection_normal` and `dpoisppLocalDetection_normal` models using the semi-complete data likelihood approach.

Usage

```
dnormalizer(x, logNormConstant, log = 0)

rnormalizer(n, logNormConstant)
```

Arguments

`x`	Input data, which can be any scalar and will not influence the return value.

`logNormConstant`	Normalizing constant on a log scale.
`log`	Logical. If `TRUE` return the log normalizing constant. Otherwise return the normalizing constant.
`n`	Integer specifying the number of realisations to generate. Only n = 1 is supported.

Value

The normalizing constant.

Author(s)

Wei Zhang

Examples

```
dnormalizer(1, log(0.5), log = TRUE)
dnormalizer(0, log(0.5), log = FALSE)
```

`dpoisLocal_normal` *Local evaluation of a Poisson SCR detection process*

Description

The `dpoisLocal_normal` distribution is a NIMBLE custom distribution which can be used to model and simulate Poisson observations (*x*) of a single individual over a set of detectors defined by their coordinates (*trapCoords*). The distribution assumes that an individual's detection probability at any detector follows a half-normal function of the distance between the individual's activity center (*s*) and the detector location.

Usage

```
dpoisLocal_normal(
  x,
  detNums = -999,
  detIndices,
  lambda = -999,
  lambdaTraps,
  sigma,
  s,
  trapCoords,
  localTrapsIndices,
  localTrapsNum,
  resizeFactor = 1,
  habitatGrid,
  indicator,
  lengthYCombined = 0,
  log = 0
)

rpoisLocal_normal(
  n = 1,
  detNums = -999,
  detIndices,
  lambda = -999,
  lambdaTraps,
  sigma,
  s,
  trapCoords,
  localTrapsIndices,
  localTrapsNum,
  resizeFactor = 1,
  habitatGrid,
  indicator,
  lengthYCombined = 0
)
```

Arguments

`x`	Vector of individual detection frequencies. This argument can be provided in two formats: (i) with the *y* object as returned by the `getSparseY` function; (ii) with the *yCombined* object as returned by `getSparseY`. Note that when the random generation functionality is used (`rpoisLocal_normal`), only the *yCombined* format can be used. The *yCombined* object combines *detNums*, *x*, and *detIndices* (in that order). When such consolidated representation of the detection data *x* is used, *detIndices* and *detNums* arguments shouldn't be specified.
`detNums`	Number of detections recorded in *x*, as returned by the *detNums* object from the `getSparseY` function. This argument should not be specified when the *yCombined* object (returned by `getSparseY`) is provided as *x*, and when detection data are simulated.
`detIndices`	Vector of indices of traps where the detections in x were recorded, as returned by the *detIndices* object from the `getSparseY` function. This argument should not be specified when *x* is provided as the *yCombined* object (returned by `getSparseY`) and when detection data are simulated.
`lambda`	Baseline detection rate used in the half-normal detection function.
`lambdaTraps`	Vector of baseline detection rate for each trap used in the half-normal detection function. When *lambdaTraps* is used, *lambda* should not be provided.
`sigma`	Scale parameter of the half-normal detection function.
`s`	Individual activity center x- and y-coordinates.
`trapCoords`	Matrix of x- and y-coordinates of all traps.

`localTrapsIndices`	Matrix of indices of local traps around each habitat grid cell, as returned by the `getLocalObjects` function.
`localTrapsNum`	Vector of numbers of local traps around all habitat grid cells, as returned by the `getLocalObjects` function.
`resizeFactor`	Aggregation factor used in the `getLocalObjects` function to reduce the number of habitat grid cells to retrieve local traps for.
`habitatGrid`	Matrix of habitat grid cells indices, as returned by the `getLocalObjects` function.
`indicator`	Logical argument specifying whether the individual is available for detection.
`lengthYCombined`	The length of the x argument when the (*yCombined*) format of the detection data is provided (as returned by the *lengthYCombined* object from `getSparseY`).
`log`	Logical argument, specifying whether to return the log-probability of the distribution.
`n`	Integer specifying the number of realizations to generate. Only `n = 1` is supported.

Details

The `dpoisLocal_normal` distribution incorporates three features to increase computation efficiency (see Turek et al., 2021 <doi.org/10.1002/ecs2.3385> for more details):

1. A local evaluation of the detection probability calculation (see Milleret et al., 2019 <doi:10.1002/ece3.4751> for more details)
2. A sparse matrix representation (*x*, *detIndices* and *detNums*) of the observation data to reduce the size of objects to be processed.
3. An indicator (*indicator*) to shortcut calculations for individuals unavailable for detection.

The `dpoisLocal_normal` distribution requires x- and y- detector coordinates (*trapCoords*) to be scaled to the habitat grid (*habitatGrid*) using the (`scaleCoordsToHabitatGrid` function.)

When the aim is to simulate detection data:

1. *x* should be provided using the *yCombined* object as returned by `getSparseY`,
2. arguments *detIndices* and *detNums* should not be provided,
3. argument *lengthYCombined* should be provided using the *lengthYCombined* object as returned by `getSparseY`.

Value

The log-likelihood value associated with the vector of detections, given the location of the activity center (s), and the half-normal detection function : $p = lambda * exp(-d^2/\sigma^2)$.

Author(s)

Cyril Milleret, Soumen Dey

Examples

```
# I. DATA SET UP
coordsHabitatGridCenter <- matrix(c(0.5, 3.5,
                                    1.5, 3.5,
                                    2.5, 3.5,
                                    3.5, 3.5,
                                    0.5, 2.5,
                                    1.5, 2.5,
                                    2.5, 2.5,
                                    3.5, 2.5,
                                    0.5, 1.5,
                                    1.5, 1.5,
                                    2.5, 1.5,
                                    3.5, 1.5,
                                    0.5, 0.5,
                                    1.5, 0.5,
                                    2.5, 0.5,
                                    3.5, 0.5), ncol=2,byrow = TRUE)
colnames(coordsHabitatGridCenter) <- c("x","y")
# CREATE OBSERVATION WINDOWS
trapCoords <- matrix(c(1.5, 1.5, 2.5, 1.5, 1.5, 2.5, 2.5, 2.5), nrow = 4, byrow = TRUE)
colnames(trapCoords) <- c("x","y")
# PLOT CHECK
plot(coordsHabitatGridCenter[,"y"]~coordsHabitatGridCenter[,"x"],pch=16)
points(trapCoords[,"y"]~trapCoords[,"x"],col="red",pch=16)

# PARAMETERS
lambda <- 0.2
sigma <- 2
indicator <- 1
# WE CONSIDER 2 INDIVIDUALS
y <- matrix(c(0, 1, 1, 0,
              0, 1, 0, 1),ncol=4,nrow=2)
s <- matrix(c(0.5, 1,
              1.6, 2.3),ncol=2,nrow=2)

# RESCALE COORDINATES
ScaledtrapCoords <- scaleCoordsToHabitatGrid(coordsData =  trapCoords,
                                      coordsHabitatGridCenter = coordsHabitatGridCenter)
ScaledtrapCoords<- ScaledtrapCoords$coordsDataScaled
habitatMask <- matrix(1, nrow = 4, ncol=4, byrow = TRUE)


# CREATE LOCAL OBJECTS
TrapLocal <- getLocalObjects(habitatMask = habitatMask,
                                   coords = ScaledtrapCoords,
                                   dmax=2.5,
                                   resizeFactor = 1,
                                   plot.check = TRUE
)

# GET SPARSE MATRIX
```

```
SparseY <- getSparseY(y)

# II. USING THE DENSITY FUNCTION
 # WE TAKE THE FIRST INDIVIDUAL
i=1
  # OPTION 1: USING THE RANDOM GENERATION FUNCTIONNALITY
dpoisLocal_normal(x=SparseY$y[i,,1],
                   detNums=SparseY$detNums[i],
                   detIndices=SparseY$detIndices[i,,1],
                   lambda = lambda,
                   sigma= sigma,
                   s=s[i,1:2],
                   trapCoords=ScaledtrapCoords,
                   localTrapsIndices=TrapLocal$localIndices,
                   localTrapsNum=TrapLocal$numLocalIndices,
                   resizeFactor=TrapLocal$resizeFactor,
                   habitatGrid=TrapLocal$habitatGrid,
                   indicator=indicator)

  # OPTION 2: USING RANDOM GENERATION FUNCTIONNALITY
  # WE DO NOT PROVIDE THE detNums AND detIndices ARGUMENTS
dpoisLocal_normal(x=SparseY$yCombined[i,,1],
                   lambda = lambda,
                   sigma= sigma,
                   s=s[i,1:2],
                   trapCoords=ScaledtrapCoords,
                   localTrapsIndices=TrapLocal$localIndices,
                   localTrapsNum=TrapLocal$numLocalIndices,
                   resizeFactor=TrapLocal$resizeFactor,
                   habitatGrid=TrapLocal$habitatGrid,
                   indicator=indicator,
                   lengthYCombined = SparseY$lengthYCombined)

# III. USING THE RANDOM GENERATION FUNCTION
rpoisLocal_normal(n=1,
                   lambda = lambda,
                   sigma= sigma,
                   s=s[i,1:2],
                   trapCoords=ScaledtrapCoords,
                   localTrapsIndices=TrapLocal$localIndices,
                   localTrapsNum=TrapLocal$numLocalIndices,
                   resizeFactor=TrapLocal$resizeFactor,
                   habitatGrid=TrapLocal$habitatGrid,
                   indicator=indicator,
                   lengthYCombined = SparseY$lengthYCombined)
```

`dpoisppAC` *Poisson point process for the distribution of activity centers*

Description

Density and random generation functions of the Poisson point process for the distribution of activity centers. The `dpoisppAC` distribution is a NIMBLE custom distribution which can be used to model and simulate activity center locations (*x*) of multiple individual in continuous space over a set of habitat windows defined by their upper and lower coordinates (*lowerCoords,upperCoords*). The distribution assumes that activity centers follow a Poisson point process with intensity = *exp(logIntensities)*.

Usage

```
dpoisppAC(
  x,
  lowerCoords,
  upperCoords,
  logIntensities,
  sumIntensity,
  habitatGrid,
  numGridRows,
  numGridCols,
  numPoints,
  log = 0
)

rpoisppAC(
  n,
  lowerCoords,
  upperCoords,
  logIntensities,
  sumIntensity,
  habitatGrid,
  numGridRows,
  numGridCols,
  numPoints
)
```

Arguments

`x`	Matrix of x- and y-coordinates of a set of spatial points (AC locations). Each row corresponds to a point.
`lowerCoords, upperCoords`	Matrices of lower and upper x- and y-coordinates of all habitat windows. One row for each window. Each window should be of size 1x1 (after rescaling if necessary).
`logIntensities`	Vector of log habitat intensities for all habitat windows.
`sumIntensity`	Sum of the habitat intensities over all windows. This can be obtained using `logIntensities`.
`habitatGrid`	Matrix of habitat window indices. Habitat window indices should match the order in `lowerCoords`, `upperCoords`, and `logIntensities`. When the grid has

	only one row/column, artificial indices have to be provided to inflate `habitatGrid` to be able to use the distribution in `nimble` model code.
`numGridRows, numGridCols`	Numbers of rows and columns of the habitat grid.
`numPoints`	Number of points in the Poisson point process. This value (non-negative integer) is used to truncate `x` so that extra rows beyond `numPoints` are ignored.
`log`	Logical argument, specifying whether to return the log-probability of the distribution.
`n`	Integer specifying the number of realisations to generate. Only n = 1 is supported.

Value

`dpoisppAC` gives the (log) probability density of the observation matrix `x`. `rpoisppAC` gives coordinates of a set of randomly generated spatial points.

Author(s)

Wei Zhang

References

W. Zhang, J. D. Chipperfield, J. B. Illian, P. Dupont, C. Milleret, P. de Valpine and R. Bischof. 2020. A hierarchical point process model for spatial capture-recapture data. bioRxiv. DOI 10.1101/2020.10.06.325035

Examples

```
lowerCoords <- matrix(c(0, 0, 1, 0, 0, 1, 1, 1), nrow = 4, byrow = TRUE)
upperCoords <- matrix(c(1, 1, 2, 1, 1, 2, 2, 2), nrow = 4, byrow = TRUE)
logIntensities <- log(c(1:4))
logSumIntensity <- sum(exp(logIntensities))
habitatGrid <- matrix(c(1:4), nrow = 2, byrow = TRUE)
numGridRows <- nrow(habitatGrid)
numGridCols <- ncol(habitatGrid)
#Simulate data
x <- rpoisppAC(1, lowerCoords, upperCoords, logIntensities, logSumIntensity, habitatGrid,
               numGridRows, numGridCols, -1)
numPoints <- nrow(x)
dpoisppAC(x, lowerCoords, upperCoords, logIntensities, logSumIntensity,
          habitatGrid, numGridRows, numGridCols, numPoints, log = TRUE)
```

dpoisppDetection_normal *Poisson point process detection model*

Description

Density and random generation functions of the Poisson point process for detection. The `dpoisppDetection_normal` distribution is a NIMBLE custom distribution which can be used to model and simulate Poisson observations (*x*) of a single individual in continuous space over a set of detection windows defined by their upper and lower coordinates (*lowerCoords,upperCoords*). The distribution assumes that an individual's detection intensity follows an isotropic multivariate normal centered on the individual's activity center (*s*) with standard deviation (*sd*).

Usage

```
dpoisppDetection_normal(
  x,
  lowerCoords,
  upperCoords,
  s,
  sd,
  baseIntensities,
  numMaxPoints,
  numWindows,
  indicator,
  log = 0
)

rpoisppDetection_normal(
  n,
  lowerCoords,
  upperCoords,
  s,
  sd,
  baseIntensities,
  numMaxPoints,
  numWindows,
  indicator
)
```

Arguments

`x`	Matrix containing the total number of detections (x[1,1]), the x- and y-coordinates (x[2:(x[1,1]+1),1:2]), and the corresponding detection window indices (x[2:(x[1,1]+1),3]) for a set of spatial points (detection locations).

`lowerCoords, upperCoords`	Matrices of lower and upper x- and y-coordinates of all detection windows. One row for each window.
`s`	Vector of x- and y-coordinates of the isotropic multivariate normal distribution mean (the AC location).
`sd`	Standard deviation of the isotropic multivariate normal distribution.
`baseIntensities`	Vector of baseline detection intensities for all detection windows.
`numMaxPoints`	Maximum number of points. This value (non-negative integer) is only used when simulating detections to constrain the maximum number of detections.
`numWindows`	Number of detection windows. This value (positive integer) is used to truncate `lowerCoords` and `upperCoords` so that extra rows beyond `numWindows` are ignored.
`indicator`	Binary variable (0 or 1) used for data augmentation. `indicator = 0` means the individual does not exist and thus the probability of no detection is 1.
`log`	Logical argument, specifying whether to return the log-probability of the distribution.
`n`	Integer specifying the number of realisations to generate. Only n = 1 is supported.

Value

`dpoisppDetection_normal` gives the (log) probability density of the observation matrix `x`. `rpoisppDetection_normal` gives coordinates of a set of randomly generated spatial points.

Author(s)

Wei Zhang

References

W. Zhang, J. D. Chipperfield, J. B. Illian, P. Dupont, C. Milleret, P. de Valpine and R. Bischof. 2020. A hierarchical point process model for spatial capture-recapture data. bioRxiv. DOI 10.1101/2020.10.06.325035

Examples

```
coordsHabitatGridCenter <- matrix(c(0.5, 3.5,
                                    1.5, 3.5,
                                    2.5, 3.5,
                                    3.5, 3.5,
                                    0.5, 2.5,
                                    1.5, 2.5,
                                    2.5, 2.5,
                                    3.5, 2.5,
                                    0.5, 1.5,
                                    1.5, 1.5,
                                    2.5, 1.5,
                                    3.5, 1.5,
                                      0.5, 0.5,
                                      1.5, 0.5,
                                      2.5, 0.5,
                                      3.5, 0.5), ncol = 2,byrow = TRUE)
colnames(coordsHabitatGridCenter) <- c("x","y")
# Create observation windows
lowerCoords <- matrix(c(0, 0, 1, 0, 0, 1, 1, 1), nrow = 4, byrow = TRUE)
upperCoords <- matrix(c(1, 1, 2, 1, 1, 2, 2, 2), nrow = 4, byrow = TRUE)
colnames(lowerCoords) <- colnames(upperCoords) <- c("x","y")

# Rescale coordinates
ScaledLowerCoords <- scaleCoordsToHabitatGrid(coordsData =  lowerCoords,
                                        coordsHabitatGridCenter = coordsHabitatGridCenter)
ScaledUpperCoords <- scaleCoordsToHabitatGrid(coordsData =  upperCoords,
                                        coordsHabitatGridCenter = coordsHabitatGridCenter)
ScaledUpperCoords$coordsDataScaled[,2] <- ScaledUpperCoords$coordsDataScaled[,2] + 1.5
ScaledLowerCoords$coordsDataScaled[,2] <- ScaledLowerCoords$coordsDataScaled[,2] - 1.5



# Detection locations
x <- matrix(c(1.5, 2, 1.1, 1.5,0.6, 2.1, 0.5, 2, 1, 1.5), nrow = 5, byrow = TRUE)

# get the window indeces on the third dimension of x

windowIndexes <- 0
for(i in 1:nrow(x)){
  windowIndexes[i] <- getWindowIndex(curCoords = x[i,],
                                     lowerCoords = ScaledLowerCoords$coordsDataScaled,
                                     upperCoords = ScaledUpperCoords$coordsDataScaled)
}
x <- cbind(x, windowIndexes)
# get the total number of detections on x[1,1]
x <- rbind(c(length(windowIndexes),0,0) ,x )


s <- c(1, 1)
sd <- 0.1
baseIntensities <- c(1:4)
windowIndices <- c(1, 2, 2, 3, 4)
numPoints <- 5
numWindows <- 4
indicator <- 1
dpoisppDetection_normal(x, lowerCoords, upperCoords, s, sd, baseIntensities,
                        numMaxPoints = dim(x)[1] , numWindows, indicator, log = TRUE)
```

dpoisppLocalDetection_normal

Local evaluation for a Poisson point process detection model

Description

Density and random generation functions of the Poisson point process for detection. The `dpoisppLocalDetection_normal` distribution is a NIMBLE custom distribution which can be used to model and simulate Poisson observations (*x*) of a single individual in continuous space over a set of detection windows defined by their upper and lower coordinates (*lowerCoords,upperCoords*). The distribution assumes that an individual's detection intensity follows an isotropic multivariate normal centered on the individual's activity center (*s*) with standard deviation (*sd*).

Usage

```
dpoisppLocalDetection_normal(
  x,
  lowerCoords,
  upperCoords,
  s,
  sd,
  baseIntensities,
  habitatGridLocal,
  resizeFactor,
  localObsWindowIndices,
  numLocalObsWindows,
  numMaxPoints,
  numWindows,
  indicator,
  log = 0
)

rpoisppLocalDetection_normal(
  n,
  lowerCoords,
  upperCoords,
  s,
  sd,
  baseIntensities,
  habitatGridLocal,
  resizeFactor,
  localObsWindowIndices,
  numLocalObsWindows,
  numMaxPoints,
  numWindows,
  indicator
)
```

Arguments

`x`	Matrix containing the total number of detections (x[1,1]), the x- and y-coordinates (x[2:(x[1,1]+1),1:2]), and the corresponding detection window indices (x[2:(x[1,1]+1),3]) for a set of spatial points (detection locations).

`lowerCoords, upperCoords`	Matrices of lower and upper x- and y-coordinates of all detection windows. One row for each window.
`s`	Vector of x- and y-coordinates of the isotropic multivariate normal distribution mean (the AC location).
`sd`	Standard deviation of the isotropic multivariate normal distribution.
`baseIntensities`	Vector of baseline detection intensities for all detection windows.
`habitatGridLocal`	Matrix of rescaled habitat grid cells indices, as returned by the `getLocalObjects` function (object named `habitatGrid`).
`resizeFactor`	Scalar (aggregation factor) for rescaling habitat windows as used in `getLocalObjects`.
`localObsWindowIndices`	Matrix of indices of local observation windows around each rescaled habitat grid cell, as returned by the getLocalObjects function (object named `localIndices`).
`numLocalObsWindows`	Vector of numbers of local observation windows around all habitat grid cells, as returned by the getLocalObjects function (object named `numLocalIndices`). The ith number gives the number of local (original) observation windows for the ith (rescaled) habitat window.
`numMaxPoints`	Maximum number of points. This value (non-negative integer) is only used when simulating detections to constrain the maximum number of detections.
`numWindows`	Number of detection windows. This value (positive integer) is used to truncate `lowerCoords` and `upperCoords` so that extra rows beyond `numWindows` are ignored.
`indicator`	Binary variable (0 or 1) used to indicate whether the individual is available for detection or not. `indicator = 0` means the individual is not available for detection (e.g. augmented or dead individual) and thus the probability of no detection is 1.
`log`	Logical argument, specifying whether to return the log-probability of the distribution.
`n`	Integer specifying the number of realisations to generate. Only n = 1 is supported.

Value

The (log) probability density of the observation matrix `x`.

Author(s)

Wei Zhang, Cyril Milleret and Pierre Dupont

References

W. Zhang, J. D. Chipperfield, J. B. Illian, P. Dupont, C. Milleret, P. de Valpine and R. Bischof. 2020.
A hierarchical point process model for spatial capture-recapture data. bioRxiv. DOI 10.1101/2020.10.06.325035

C. Milleret, P. Dupont, C. Bonenfant, H. Brøseth, Ø. Flagstad, C. Sutherland and R. Bischof. 2019. A local evaluation of the individual state-space to scale up Bayesian spatial capture-recapture. Ecology and Evolution 9:352-363

#' @examples # Create habitat grid coordsHabitatGridCenter <- matrix(c(0.5, 3.5, 1.5, 3.5, 2.5, 3.5, 3.5, 3.5, 0.5, 2.5, 1.5, 2.5, 2.5, 2.5, 3.5, 2.5, 0.5, 1.5, 1.5, 1.5, 2.5, 1.5, 3.5, 1.5, 0.5, 0.5, 1.5, 0.5, 2.5, 0.5, 3.5, 0.5), ncol = 2, byrow = TRUE) colnames(coordsHabitatGridCenter) <- c("x","y") # Create observation windows lowerCoords <- matrix(c(1, 1, 2, 1, 1, 2, 2, 2), nrow = 4, byrow = TRUE) upperCoords <- matrix(c(2, 2, 3, 2, 2, 3, 3, 3), nrow = 4, byrow = TRUE) colnames(lowerCoords) <- colnames(upperCoords) <- c("x","y") # Plot check plot(coordsHabitatGridCenter[,"y"]~coordsHabitatGridCenter[,"x"],pch= points(lowerCoords[,"y"]~lowerCoords[,"x"],col="red",pch=16) points(upperCoords[,"y"]~upperCoords[,"x"],col="red",pch

s <- c(1, 1) sd <- 0.1 baseIntensities <- c(1:4) windowIndex <- 4 numPoints <- 1 numWindows <- 4 indicator <- 1

# Rescale coordinates ScaledLowerCoords <- scaleCoordsToHabitatGrid(coordsData = lowerCoords, coordsHabitatGridCenter = coordsHabitatGridCenter)$coordsDataScaled ScaledUpperCoords <- scaleCoordsToHabitatGrid(coordsData = upperCoords, coordsHabitatGridCenter = coordsHabitatGridCenter)$coordsDataScaled ScaledUpperCoords[,2] <- ScaledUpperCoords[,2] + 1.5 ScaledLowerCoords[,2] <- ScaledLowerCoords[,2] - 1.5 habitatMask <- matrix(1, nrow = 4, ncol=4, byrow = TRUE) # Create local objects ObsWindowsLocal <- getLocalObjects(habitatMask = habitatMask, coords = ScaledLowerCoords, dmax=3, resizeFactor = 1, plot.check = TRUE)

# Detection locations x <- matrix(c(1.5, 2, 1.1, 1.5, 1.4, 0.7, 2, 1.3, 1, 1.5), nrow = 5, byrow = TRUE)

# get the window indeces on the third dimension of x windowIndexes <- 0 for(i in 1:nrow(x)) windowIndexes[i] <- getWindowIndex(curCoords = x[i,], lowerCoords = ScaledLowerCoords, upperCoords =ScaledUpperCoords)

x <- cbind(x, windowIndexes) # get the total number of detections on x[1,1] x <- rbind(c(length(windowIndexes),0,0), x) dpoisppLocalDetection_normal(x, ScaledLowerCoords, ScaledUpperCoords, s, sd, baseIntensities, ObsWindowsLocal$habitatGrid, ObsWindowsLocal$resizeFactor, ObsWindowsLocal$localIndices,ObsWindowsLocal$ numMaxPoints = dim(x)[1], numWindows, indicator, log = TRUE)

`getLocalObjects` *Local Objects Identification*

Description

R utility function to identify all objects (e.g. traps) within a given radius dmax of each cell in a habitat mask. Used in the implementation of the local evaluation approach in SCR models (`dbinomLocal_normal`;`dpoisLocal_normal`). The distance to the activity center and the detection probability are then calculated for local objects only (i.e. the detection probability is assumed to be 0 for all other objects as they are far enough from the activity center).

Usage

```
getLocalObjects(habitatMask, coords, dmax, resizeFactor = 1, plot.check = TRUE)
```

Arguments

`habitatMask`	a binary matrix object indicating which cells are considered as suitable habitat.
`coords`	A matrix giving the x- and y-coordinate of each object (i.e. trap). x- and y-coordinates should be scaled to the habitat (`scaleCoordsToHabitatGrid`).
`dmax`	The maximal radius from a habitat cell center within which detection probability is evaluated locally for each trap.
`resizeFactor`	An aggregation factor to reduce the number of habitat cells to retrieve local objects for. Defaults to 1; no aggregation.
`plot.check`	A visualization option (if TRUE); displays which objects are considered "local objects" for a randomly chosen habitat cell.

Details

The `getLocalObjects` function is used in advance of model building.

Value

This function returns a list of objects:

- localIndices: a matrix with number of rows equal to the reduced number of habitat grid cells (following aggregation). Each row gives the id numbers of the local objects associated with this grid cell.
- habitatGrid: a matrix of habitat grid cells ID corresponding to the row indices in localIndices.
- numLocalIndices: a vector of the number of local objects for each habitat grid cell in habitatGrid.
- numLocalIndicesMax: the maximum number of local objects for any habitat grid cell ; corresponds to the number of columns in habitatGrid.
- resizeFactor: the aggregation factor used to reduce the number of habitat grid cells.

Author(s)

Cyril Milleret and Pierre Dupont

Examples

```
colNum <- sample(20:100,1)
rowNum <- sample(20:100,1)
coords <- expand.grid(list(x = seq(0.5, colNum, 1),
                           y = seq(0.5, rowNum, 1)))

habitatMask <- matrix(rbinom(colNum*rowNum, 1, 0.8), ncol = colNum, nrow = rowNum)

localObject.list <- getLocalObjects(habitatMask, coords,  dmax = 7,resizeFactor = 1)
```

`getLocalTraps`	*Local Trap Identification. This function is deprecated, use* `getLocalObjects` *instead.*

Description

R utility function to identify all traps within a given radius dmax of each cell in a habitat mask. Used in the implementation of the local evaluation approach in SCR models (`dbinomLocal_normal`). The distance to the activity center and the detection probability are then calculated for these local traps only (i.e. the detection probability is assumed to be 0 for all other traps as they are far enough from the activity center).

Usage

```
getLocalTraps(
  habitatMask,
  trapCoords,
  dmax,
  resizeFactor = 1,
  plot.check = TRUE
)
```

Arguments

`habitatMask`	a binary matrix object indicating which cells are considered as suitable habitat.
`trapCoords`	A matrix giving the x- and y-coordinate of each trap.
`dmax`	The maximal radius from a habitat cell center within which detection probability is evaluated locally for each trap.
`resizeFactor`	An aggregation factor to reduce the number of habitat cells to retrieve local traps for. Defaults to 1; no aggregation.
`plot.check`	A visualization option (if TRUE); displays which traps are considered "local traps" for a randomly chosen habitat cell.

Details

The `getLocalTraps` function is used in advance of model building.

Value

This function returns a list of objects:

- localTrapIndices: a matrix with number of rows equal to the reduced number of habitat grid cells (following aggregation). Each row gives the id numbers of the local traps associated with this grid cell.
- habitatGrid: a matrix of habitat grid cells ID corresponding to the row indices in localTrapIndices.

- numLocalTraps: a vector of the number of local traps for each habitat grid cell in habitatGrid.
- numLocalTrapsMax: the maximum number of local traps for any habitat grid cell ; corresponds to the number of columns in habitatGrid.
- resizeFactor: the aggregation factor used to reduce the number of habitat grid cells.

Author(s)

Cyril Milleret and Pierre Dupont

Examples

```
colNum <- sample(20:100,1)
rowNum <- sample(20:100,1)
trapCoords <- expand.grid(list(x = seq(0.5, colNum, 1),
                               y = seq(0.5, rowNum, 1)))

habitatMask <- matrix(rbinom(colNum*rowNum, 1, 0.8), ncol = colNum, nrow = rowNum)

localTraps.list <- getLocalTraps(habitatMask, trapCoords, resizeFactor = 1, dmax = 7)
```

`getMidPointNodes` *Generate midpoint integration nodes*

Description

Generate midpoint nodes and weights for integrating a function numerically over a set of windows. For each window, generate a set of equally spaced nodes and weights.

Usage

```
getMidPointNodes(lowerCoords, upperCoords, numSubintervals = 10)
```

Arguments

`lowerCoords, upperCoords`	Matrices of lower and upper x- and y-coordinates of a set of windows. One row for each window.
`numSubintervals`	Number of subintervals each dimension of a window is divided into.

Value

A list of midpoint nodes and weights.

Author(s)

Wei Zhang

Examples

```
lowerCoords <- matrix(c(0, 0, 1, 0, 0, 1, 1, 1), nrow = 4, byrow = TRUE)
upperCoords <- matrix(c(1, 1, 2, 1, 1, 2, 2, 2), nrow = 4, byrow = TRUE)
getMidPointNodes(lowerCoords, upperCoords, 5)
```

`getSparseY` *Sparse Matrix Preparation*

Description

R utility function to turn a two or three-dimensional detection array into a sparse matrix representation (see Turek et al., 2021 <doi.org/10.1002/ecs2.3385> for more details). Used in the implementation of the `dbinomLocal_normal` and `dpoisLocal_normal` functions.

Usage

```
getSparseY(x, noDetections = -1, nMaxTraps = NULL)
```

Arguments

`x`	A two- or three-dimensional observation data array with dimensions : number of individuals, number of traps, (and number of detection occasions/sessions).
`noDetections`	The value indicating no detection. Defaults to -1.
`nMaxTraps`	The maximum number of traps at which detections can occur. It is necessary to artificially augment the sparse detection array when using the random generation functionality of the dbinomLocal_normal or dpoisLocal_normal functions. When simulating detection data, augmenting the size of the detection array is necessary to avoids artificially limiting the number of detectors at which individuals can be detected. Default value is maxDetNums * 2, which doubles the maximum number of traps at which an individual can be detected. We generally recommend using *numLocalIndicesMax* obtained from `getLocalObjects` when aiming at randomly generating detections from dbinomLocal_normal or dpoisLocal_normal.

Details

The `getSparseY` function is used in advance of model building to create a sparse matrix representation of the observation data. It creates and returns a list of objects:

Value

A list of objects which constitute a sparse representation of the observation data:

- *detNums* A matrix with number of traps at which each individual (in rows) was detected at each occasions/sessions (in columns).

- *maxDetNums* The maximum number of traps at which an individual was detected (i.e., the maximum of *detNums*).
- *detIndices* An array of dimensions n.individuals, maxDetNums, and number of occasions/sessions, which contains the IDs of the traps where each individual was detected.
- *y* An array of dimensions n.individuals, maxDetNums, and occasions/sessions, which contains the number of observations of each individual at the traps it was detected at.
- *yCombined* An array that combines *detNums*, *y*, and *detIndices* by columns (in that specific order). Note that *y*, and *detIndices* are augmented before combining, such that the maximum number of detectors at which an individual can be detected is equal to *nMaxTraps* Consequently, the number of columns of *lengthYCombined* is 2*nMaxTraps + 1.
- *lengthYCombined* Dimension of the augmented lengthYCombined object to be specified as the argument *lengthYCombined* of the `dbinomLocal_normal` or `dpoisLocal_normal` functions when simulating detection data.

Author(s)

Cyril Milleret

Examples

```
y.full <- matrix(rbinom(5000, 5, 0.02), ncol = 100)

y <- getSparseY(y.full)
```

getWindowCoords *Get lower and upper windows coordinates*

Description

The `getWindowCoords` is an R utility function to create lower and upper habitat and observation windows coordinates, as well an habitat grid with cell ids. Those objects are necessary to run all point process (pp) functions. All input data should be scaled to the habitat grid using `scaleCoordsToHabitatGrid`. Note that we assume homogeneous window sizes.

Usage

```
getWindowCoords(
  scaledHabGridCenter = scaledHabGridCenter,
  scaledObsGridCenter = NULL,
  plot.check = TRUE
)
```

Arguments

`scaledHabGridCenter`	A matrix with the scaled "x" and "y" habitat windows grid cell centers (after using `scaleCoordsToHabitatGrid`).
`scaledObsGridCenter`	A matrix with the scaled "x" and "y" observation windows grid cell centers (afer using `scaleCoordsToHabitatGrid`). This is an optional argument and only necessary when modelling detection as a point process (e.g. `dpoisppDetection_normal`).
`plot.check`	A visualization option (if TRUE); displays habitat and detection windows.

Value

A list of objects :

- *lowerHabCoords* A matrix with the "x" and "y" lower habitat window coordinates.
- *upperHabCoords* A matrix with the "x" and "y" upper habitat window coordinates.
- *habitatGrid* A matrix of habitat cell ID that can be used to lookup efficiently the cell ID from a coordinate scaled to the habitat grid: habitatGrid[trunc(scaledHabGridCenter[1,"y"]) + 1, trunc(scaledHabGridCenter[1,"x"]) + 1]. See `scaleCoordsToHabitatGrid` for more details.
- *lowerObsCoords* A matrix with the "x" and "y" lower observation window coordinates. Only returned when *scaledObsGridCenter* is provided.
- *upperObsCoords* A matrix with the "x" and "y" upper observation window coordinates. Only returned when *scaledObsGridCenter* is provided.

Author(s)

Cyril Milleret

Examples

```
coordsGridCenter <- expand.grid(list(x = seq(50.5, 100, 1),
                                     y = seq(100.5, 150, 1)))
coordsData <- expand.grid(list(x = seq(60, 90, 1),
                               y = seq(110, 140, 1)))

plot(coordsGridCenter[,2] ~ coordsGridCenter[,1])
points(coordsData[,2] ~ coordsData[,1], col="red")
scaled <- scaleCoordsToHabitatGrid(coordsData = coordsData
                                   , coordsHabitatGridCenter = coordsGridCenter)
plot(scaled$coordsHabitatGridCenterScaled[,2] ~ scaled$coordsHabitatGridCenterScaled[,1])
points(scaled$coordsDataScaled[,2] ~ scaled$coordsDataScaled[,1], col="red")

LowerAndUpperCoords <- getWindowCoords(scaledHabGridCenter = scaled$coordsHabitatGridCenterScaled,
                                         scaledObsGridCenter = scaled$coordsDataScaled)

# Plot habitat window cell centers and lower/upper coordinates
plot(scaled$coordsHabitatGridCenterScaled[,2] ~
     scaled$coordsHabitatGridCenterScaled[,1],
     pch=16, cex=0.3, col=grey(0.5))
```

```
points(LowerAndUpperCoords$lowerHabCoords[,2] ~
       LowerAndUpperCoords$lowerHabCoords[,1],
       pch=16, cex=0.3, col=grey(0.1))
points(LowerAndUpperCoords$upperHabCoords[,2] ~
       LowerAndUpperCoords$upperHabCoords[,1],
       pch=16, cex=0.3, col=grey(0.1))

# Plot observation window cells center and lower/upper coordinates
points(scaled$coordsDataScaled[,2]~scaled$coordsDataScaled[,1], pch=16,
 cex=0.3, col = adjustcolor("red",alpha.f = 0.8))
points(LowerAndUpperCoords$lowerObsCoords[,2] ~
        LowerAndUpperCoords$lowerObsCoords[,1],
       pch=16, cex=0.3, col = adjustcolor("red", alpha.f = 0.8))
points(LowerAndUpperCoords$upperObsCoords[,2] ~
       LowerAndUpperCoords$upperObsCoords[,1],
       pch=16, cex=0.3, col = adjustcolor("red", alpha.f = 0.8))
```

`getWindowIndex` *Get window index*

Description

From a set of windows, find the index of the window into which a given point falls. Can be applied to detection and habitat windows.

Usage

```
getWindowIndex(curCoords, lowerCoords, upperCoords)
```

Arguments

`curCoords`	Vector of coordinates of a single spatial point
`lowerCoords, upperCoords`	Matrices of lower and upper x- and y-coordinates of a set of windows. One row for each window.

Value

Index of the window where the given point falls; -1 is returned if the point does not fall in any window.

Author(s)

Pierre Dupont

Examples

```
sourceCoords <- c(1.5,2.2)
lowerCoords <- cbind(c(0,1,3,0),c(0,1,2,2))
upperCoords <- cbind(c(1,3,5,3),c(1,2,4,4))
getWindowIndex(sourceCoords, lowerCoords, upperCoords)
```

`integrateIntensityLocal_normal`
Integrate the multivariate normal intensity with local evaluation

---

Description

Calculate the integral of the intensity function with an isotropic multivariate normal kernel over a set of windows. The local evaluation technique is implemented.

Usage

```
integrateIntensityLocal_normal(
  lowerCoords,
  upperCoords,
  s,
  baseIntensities,
  sd,
  numLocalWindows,
  localWindows
)
```

Arguments

`lowerCoords, upperCoords`	Matrices of lower and upper x- and y-coordinates of a set of windows. One row for each window.
`s`	Vector of x- and y-coordinates of the isotropic multivariate normal distribution mean (AC location).
`baseIntensities`	Vector of baseline intensities for all windows.
`sd`	Standard deviation of the isotropic multivariate normal distribution.
`numLocalWindows`	Number of windows that are close to the activity center
`localWindows`	Vector of indices of the windows that are close to the activity center.

Value

A vector of integrated intensities over all local windows.

Author(s)

Cyril Milleret and Wei Zhang

References

W. Zhang, J. D. Chipperfield, J. B. Illian, P. Dupont, C. Milleret, P. de Valpine and R. Bischof. 2020. A hierarchical point process model for spatial capture-recapture data. bioRxiv. DOI 10.1101/2020.10.06.325035

Examples

```
lowerCoords <- matrix(c(0, 0, 1, 0, 0, 1, 1, 1), nrow = 4, byrow = TRUE)
upperCoords <- matrix(c(1, 1, 2, 1, 1, 2, 2, 2), nrow = 4, byrow = TRUE)
s <- c(0.1, 0.9)
sd <- 0.1
baseIntensities <- c(1:4)
numLocalWindows <- 2
localWindows <- c(1, 3)
integrateIntensityLocal_normal(lowerCoords, upperCoords, s,
                               baseIntensities, sd,
                               numLocalWindows, localWindows)
```

---

integrateIntensity_exp

Integrate the multivariate exponential intensity

Description

Calculate the integral of the intensity function with an isotropic multivariate exponential kernel over a set of windows.

Usage

```
integrateIntensity_exp(
  lowerCoords,
  upperCoords,
  s,
  baseIntensities,
  lambda,
  numWindows
)
```

Arguments

`lowerCoords, upperCoords`	Matrices of lower and upper x- and y-coordinates of a set of windows. One row for each window.

`s`	Vector of x- and y-coordinates of the AC location.
`baseIntensities`	Vector of baseline intensities for all windows.
`lambda`	Rate parameter of the isotropic multivariate exponential distribution.
`numWindows`	Total number of windows. This value (positive integer) is used to truncate `lowerCoords` and `upperCoords` so that extra rows beyond `numWindows` are ignored.

Value

A vector of integrated intensities over all windows.

Author(s)

Wei Zhang

References

W. Zhang, J. D. Chipperfield, J. B. Illian, P. Dupont, C. Milleret, P. de Valpine and R. Bischof. 2020. A hierarchical point process model for spatial capture-recapture data. bioRxiv. DOI 10.1101/2020.10.06.325035

Examples

```
lowerCoords <- matrix(c(0, 0, 1, 0, 0, 1, 1, 1), nrow = 4, byrow = TRUE)
upperCoords <- matrix(c(1, 1, 2, 1, 1, 2, 2, 2), nrow = 4, byrow = TRUE)
s <- c(1, 1)
lambda <- 1.0
baseIntensities <- c(1:4)
numWindows <- 4
integrateIntensity_exp(lowerCoords, upperCoords, s, baseIntensities, lambda, numWindows)
```

`integrateIntensity_normal` *Integrate the multivariate normal intensity*

Description

Calculate the integral of the intensity function with an isotropic multivariate normal kernel over a set of windows.

Usage

```
integrateIntensity_normal(
  lowerCoords,
  upperCoords,
  s,
  baseIntensities,
  sd,
  numWindows
)
```

Arguments

`lowerCoords, upperCoords`	Matrices of lower and upper x- and y-coordinates of a set of windows. One row for each window.
`s`	Vector of x- and y-coordinates of the isotropic multivariate normal distribution mean (AC location).
`baseIntensities`	Vector of baseline intensities for all windows.
`sd`	Standard deviation of the isotropic multivariate normal distribution.
`numWindows`	Total number of windows. This value (positive integer) is used to truncate `lowerCoords` and `upperCoords` so that extra rows beyond `numWindows` are ignored.

Value

A vector of integrated intensities over all windows.

Author(s)

Wei Zhang

References

W. Zhang, J. D. Chipperfield, J. B. Illian, P. Dupont, C. Milleret, P. de Valpine and R. Bischof. 2020.
A hierarchical point process model for spatial capture-recapture data. bioRxiv. DOI 10.1101/2020.10.06.325035

Examples

```
lowerCoords <- matrix(c(0, 0, 1, 0, 0, 1, 1, 1), nrow = 4, byrow = TRUE)
upperCoords <- matrix(c(1, 1, 2, 1, 1, 2, 2, 2), nrow = 4, byrow = TRUE)
s <- c(1, 1)
sd <- 0.1
baseIntensities <- c(1:4)
numWindows <- 4
integrateIntensity_normal(lowerCoords, upperCoords, s, baseIntensities, sd, numWindows)
```

`localTrapCalculations` *Local Trap Calculations*

Description

Utility functions to enable local trap calculations in SCR models. See details section for more information.

Usage

```
makeGrid(xmin = 0, ymin = 0, xmax, ymax, resolution = 1, buffer = 0)

findLocalTraps(grid, trapCoords, dmax)

getNumLocalTraps(idarg, nLocalTraps, LTD1arg)

getLocalTrapIndices(MAXNUM, localTraps, n, idarg)

calcLocalTrapDists(MAXNUM, n, localTrapInd, s, trapCoords)

calcLocalTrapExposure(R, n, d, localTrapInd, sigma, p0)
```

Arguments

`xmin`	Minimal value among all trap location x-coordinates.
`ymin`	Minimal value among all trap location y-coordinates.
`xmax`	Maximal value among all trap location x-coordinates.
`ymax`	Maximal value among all trap location y-coordinates.
`resolution`	Desired resolution (in both x and y directions) of discretized grid.
`buffer`	Horizontal and vertical buffer for discretized grid, specifying how much it should extend (above, below, left, and right) of the maximal trap locations.
`grid`	The grid object returned from the makeGrid function.
`trapCoords`	An nTraps x 2 array giving giving the x- and y-coordinate locations of all traps.
`dmax`	The maximal radius from an activity center for performing trap calculations (dmax).
`idarg`	A grid id, returned from the makeID function inside model code.
`nLocalTraps`	The number of local traps to all grid cells, which is given by the first column of the localTraps array.
`LTD1arg`	The number of columns in the localTraps array.
`MAXNUM`	The maximum number of local traps among all grid cells. This is given by the (number of rows)-1 of the localTraps array.
`localTraps`	The array returned from the findLocalTraps function.
`n`	The number of local traps to a specified grid cell, as return.

`localTrapInd`	The indices of the local traps to a grid cell, as returned by the getLocalTrapIndices function.
`s`	A length-2 vector giving the activity center of an indiviual.
`R`	The total number of traps.
`d`	A vector of distances from an activity center to the local traps.
`sigma`	Scale of decay for detection probability.
`p0`	Baseline detection probability.

Details

The makeGrid function is used in advance of model building. It creates and returns a list of two objects: a table (grid) corresponding to the discretized grid, where each row gives the x-coordinate, the y-coordinate, and the id number for a grid cell; and second, a function (makeID) to be used in the model code which operates on a discretized AC location, and returns the id number of the corresponding grid cell.

The findLocalTraps function operates on the grid object returned from makeGrid, and an array of the trap location coordinates, and the desired maximal exposure radius for caluclations (dmax). It returns a array (localTraps) with number of rows equal to the number of grid cells. The first element of each row gives the number of local traps within exposure radius to that grid cell. The following elements of each row give the id numbers of those local traps.

A visualization function (plotTraps) is also provided in the example code, which displaces the discretized grid (small black points), all trap locations (green circles), a specified grid cell location (specified by i) as a large X, and the local traps to that specified grid cell (red circles).

The getNumLocalTraps function is used inside the model code. It operates on an id for a grid cell, the localTraps array (generated by findLocalTraps), and the constant value LTD1. This function returns the number of traps which are local to a specified grid cell.

The getLocalTrapIndices function is used inside the model code. It returns a vector containing the ids of the local traps to a particular grid cell.

The calcLocalTrapDists function is used inside the model code. It calculates the distances from an activity center, to the local traps relative to the grid cell nearest that activity center.

The calcLocalTrapExposure function is specific to the detection probability calculations used in this example. This function should be modified specifically to the detection function, exposure function, or otherwise calculations to be done only for the traps in the vicinity of individual activity center locations

Author(s)

Daniel Turek

Examples

```
## generate random trap locations
nTraps <- 200
traps_xmin <- 0
traps_ymin <- 0
traps_xmax <- 100
```

```
traps_ymax <- 200
set.seed(0)
traps_xCoords <- round(runif(nTraps, traps_xmin, traps_xmax))
traps_yCoords <- round(runif(nTraps, traps_ymin, traps_ymax))
trap_coords <- cbind(traps_xCoords, traps_yCoords)

## buffer distance surrounding sides of rectangular discretization grid
## which overlays trap locations
buffer <- 10

## resolution of rectangular discretization grid
resolution <- 10

## creates grid and makeID function,
## for grid overlaying trap locations,
## and to lookup nearest grid cell to any AC
makeGridReturn <- makeGrid(xmin = traps_xmin, xmax = traps_xmax,
                           ymin = traps_ymin, ymax = traps_ymax,
                           buffer = buffer,
                           resolution = resolution)

grid <- makeGridReturn$grid
makeID <- makeGridReturn$makeID

## maximum radis within an individual AC to perform trap calculations,
dmax <- 30

## n = localTraps[i,1] gives the number of local traps
## localTraps[i, 2:(n+1)] gives the indices of the local traps
localTraps <- findLocalTraps(grid, trap_coords, dmax)

plotTraps <- function(i, grid, trap_coords, localTraps) {
    plot(grid[,1], grid[,2], pch = '.', cex=2)
    points(trap_coords[,1], trap_coords[,2], pch=20, col='forestgreen', cex=1)
    if(!missing(i)) {
        i <- max(i %% dim(grid)[1], 1)
        n <- localTraps[i,1]
        trapInd <- numeric(0)
        if(n > 0)  trapInd <- localTraps[i,2:(n+1)]
        theseTraps <- trap_coords[trapInd,, drop = FALSE]
        points(theseTraps[,1], theseTraps[,2], pch = 20, col = 'red', cex=1.5)
        points(grid[i,1], grid[i,2], pch = 'x', col = 'blue', cex=3)
    }
}

## visualise some local traps
plotTraps(10,  grid, trap_coords, localTraps)
plotTraps(200, grid, trap_coords, localTraps)
plotTraps(380, grid, trap_coords, localTraps)

## example model code
## using local trap calculations
code <- nimbleCode({
    sigma ~ dunif(0, 100)
    p0 ~ dunif(0, 1)
    for(i in 1:N) {
        S[i,1] ~ dunif(0, xmax)
        S[i,2] ~ dunif(0, ymax)
        Sdiscrete[i,1] <- round(S[i,1]/res) * res
        Sdiscrete[i,2] <- round(S[i,2]/res) * res
        id[i] <- makeID( Sdiscrete[i,1:2] )
        nLocalTraps[i] <- getNumLocalTraps(id[i], localTraps[1:LTD1,1], LTD1)
        localTrapIndices[i,1:maxTraps] <-
         getLocalTrapIndices(maxTraps, localTraps[1:LTD1,1:LTD2], nLocalTraps[i], id[i])
        d[i, 1:maxTraps] <- calcLocalTrapDists(
            maxTraps, nLocalTraps[i], localTrapIndices[i,1:maxTraps],
            S[i,1:2], trap_coords[1:nTraps,1:2])
        g[i, 1:nTraps] <- calcLocalTrapExposure(
        nTraps, nLocalTraps[i], d[i,1:maxTraps], localTrapIndices[i,1:maxTraps], sigma, p0)
        y[i, 1:nTraps] ~ dbinom_vector(prob = g[i,1:nTraps], size = trials[1:nTraps])
    }
})

## generate random detection data; completely random
N <- 100
set.seed(0)
y <- array(rbinom(N*nTraps, size=1, prob=0.8), c(N, nTraps))

## generate AC location initial values
Sinit <- cbind(runif(N, traps_xmin, traps_xmax),
               runif(N, traps_ymin, traps_ymax))

constants <- list(N = N,
                  nTraps = nTraps,
                  trap_coords = trap_coords,
                  xmax = traps_xmax,
                  ymax = traps_ymax,
                  res = resolution,
                  localTraps = localTraps,
                  LTD1 = dim(localTraps)[1],
                  LTD2 = dim(localTraps)[2],
                  maxTraps = dim(localTraps)[2] - 1)

data <- list(y = y, trials = rep(1,nTraps))

inits <- list(sigma = 1,
              p0 = 0.5,
              S = Sinit)

## create NIMBLE model object
Rmodel <- nimbleModel(code, constants, data, inits,
                      calculate = FALSE, check = FALSE)

## use model object for MCMC, etc.
```

`marginalVoidProbIntegrand` *Integrand of the marginal void probability integral*

Description

Integrand of the marginal void probability integral. The domain of this function is the habitat domain.

Usage

```
marginalVoidProbIntegrand(
  x,
  lowerCoords,
  upperCoords,
  sd,
  baseIntensities,
  numPoints,
  numWindows
)
```

Arguments

`x`	Matrix of x- and y-coordinates of a set of spatial points. One row corresponds to one point.
`lowerCoords, upperCoords`	Matrices of lower and upper x- and y-coordinates of a set of detection windows. One row for each window.
`sd`	Standard deviation of the isotropic multivariate normal distribution.
`baseIntensities`	Vector of baseline detection intensities for all detection windows.
`numPoints`	Number of points that should be considered. This value (positive integer) is used to truncate `x` so that extra rows beyond `numPoints` are ignored.
`numWindows`	Number of windows. This value (positive integer) is used to truncate `lowerCoords` and `upperCoords` so that extra rows beyond `numWindows` are ignored.

Value

A vector of values of the integrand evaluated at each point of `x`.

Author(s)

Wei Zhang

References

W. Zhang, J. D. Chipperfield, J. B. Illian, P. Dupont, C. Milleret, P. de Valpine and R. Bischof. 2020. A hierarchical point process model for spatial capture-recapture data. bioRxiv. DOI 10.1101/2020.10.06.325035

Examples

```
x <- matrix(runif(10, 0, 2), nrow = 5)
lowerCoords <- matrix(c(0, 0, 1, 0, 0, 1, 1, 1), nrow = 4, byrow = TRUE)
upperCoords <- matrix(c(1, 1, 2, 1, 1, 2, 2, 2), nrow = 4, byrow = TRUE)
sd <- 0.1
baseIntensities <- c(1:4)
numPoints <- 5
numWindows <- 4
marginalVoidProbIntegrand(x, lowerCoords, upperCoords, sd, baseIntensities, numPoints, numWindows)
```

---

marginalVoidProbNumIntegration *Marginal void probability*

Description

Calculate the marginal void probability using the midpoint integration method.

Usage

```
marginalVoidProbNumIntegration(
  quadNodes,
  quadWeights,
  numNodes,
  lowerCoords,
  upperCoords,
  sd,
  baseIntensities,
  habIntensities,
  sumHabIntensity,
  numObsWindows,
  numHabWindows
)
```

Arguments

`quadNodes`	Three-dimensional array of nodes for midpoint integration. The dimension sizes are equal to the number of nodes per habitat window (1st), 2 (2nd), and the number of habitat windows (3rd).
`quadWeights`	Vector of weights for midpoint integration.
`numNodes`	Vector of numbers of nodes for all habitat windows.

`lowerCoords, upperCoords`	Matrix of lower and upper x- and y-coordinates of all detection windows. One row for each window.
`sd`	Standard deviation of the isotropic multivariate normal distribution.
`baseIntensities`	Vector of baseline detection intensities for all detection windows.
`habIntensities`	Vector of habitat intensities for all habitat windows.
`sumHabIntensity`	Total habitat selection intensity over all windows.
`numObsWindows`	Number of detection windows.
`numHabWindows`	Number of habitat windows.

Value

The marginal void probability.

Author(s)

Wei Zhang

References

W. Zhang, J. D. Chipperfield, J. B. Illian, P. Dupont, C. Milleret, P. de Valpine and R. Bischof. 2020. A hierarchical point process model for spatial capture-recapture data. bioRxiv. DOI 10.1101/2020.10.06.325035

Examples

```
lowerHabCoords <- matrix(c(0, 0, 0, 1), nrow = 2, byrow = TRUE)
upperHabCoords <- matrix(c(2, 1, 2, 2), nrow = 2, byrow = TRUE)
lowerObsCoords <- matrix(c(0, 0, 1, 0, 0, 1, 1, 1), nrow = 4, byrow = TRUE)
upperObsCoords <- matrix(c(1, 1, 2, 1, 1, 2, 2, 2), nrow = 4, byrow = TRUE)
nodesRes <- getMidPointNodes(lowerHabCoords, upperHabCoords, 10)
quadNodes <- nodesRes$quadNodes
quadWeights <- nodesRes$quadWeights
numNodes <- rep(100, 2)
sd <- 0.1
baseDetIntensities <- c(1:4)
habIntensities <- c(1:2)
sumHabIntensity <- sum(habIntensities * c(2, 2))
numObsWindows <- 4
numHabWindows <- 2
marginalVoidProbNumIntegration(quadNodes, quadWeights, numNodes,
                               lowerObsCoords, upperObsCoords, sd,
                               baseDetIntensities, habIntensities,
                               sumHabIntensity, numObsWindows, numHabWindows)
```

scaleCoordsToHabitatGrid

Scale x- and y-coordinates to grid cells coordinates.

Description

R utility function to scale x- and y- coordinates to the habitat grid. Scaling the coordinates to the habitat grid allows implementation of the fast look-up approach to identify the habitat grid cell in which a point is located. This technique was first applied by Mike Meredith in SCR (https://mmeredith.net/blog/2013/1309_SECR_in_JAGS_patchy_habitat.htm). Re-scaling the entire coordinate system of the data input is a requirement to run SCR models with the local evaluation approach. This function requires square grid cells and coordinates using projection with units in meters or km (e.g., UTM but not latitude/longitude)

Usage

```
scaleCoordsToHabitatGrid(
  coordsData = coordsData,
  coordsHabitatGridCenter = coordsHabitatGridCenter,
  scaleToGrid = TRUE
)
```

Arguments

`coordsData`	A matrix or array of x- and y-coordinates to be scaled to the habitat grid. x- and y- coordinates must be identified using "x" and "y" dimnames.
`coordsHabitatGridCenter`	A matrix of x- and y-coordinates for each habitat grid cell center.
`scaleToGrid`	Defaults to TRUE. If FALSE, coordsData are already scaled and will be rescaled to its original coordinates.

Value

This function returns a list of objects:

- coordsDataScaled: A matrix or array of scaled (rescaled if scaleToGrid==FALSE) x- and y-coordinates for coordsData.
- coordsHabitatGridCenterScaled: A matrix of scaled x- and y-cell coordinates for coordsHabitatGridCenter.

Author(s)

Richard Bischof, Cyril Milleret

Examples

```
coordsGridCenter <- expand.grid(list(x = seq(50.5, 100, 1),
                                     y = seq(100.5, 150, 1)))
coordsData <- expand.grid(list(x = seq(60, 90, 1),
                              y = seq(110, 140, 1)))
plot(coordsGridCenter[,2]~coordsGridCenter[,1])
points(coordsData[,2]~coordsData[,1], col="red")
scaled <- scaleCoordsToHabitatGrid(coordsData = coordsData
                                  , coordsHabitatGridCenter = coordsGridCenter)
plot(scaled$coordsHabitatGridCenterScaled[,2]~scaled$coordsHabitatGridCenterScaled[,1])
points(scaled$coordsDataScaled[,2]~scaled$coordsDataScaled[,1], col="red")
```

---

stratRejectionSampler_exp

Stratified rejection sampler for multivariate exponential point process

---

Description

Simulate data using a stratified rejection sampler from a point process with an isotropic multivariate exponential decay kernel.

Usage

```
stratRejectionSampler_exp(
  numPoints,
  lowerCoords,
  upperCoords,
  s,
  windowIntensities,
  lambda
)
```

Arguments

`numPoints`	Number of spatial points to generate.
`lowerCoords, upperCoords`	Matrices of lower and upper x- and y-coordinates of a set of detection windows. One row for each window.
`s`	Vector of x- and y-coordinates of of the isotropic multivariate exponential distribution mean.
`windowIntensities`	Vector of integrated intensities over all detection windows.
`lambda`	Rate parameter of the isotropic multivariate exponential distribution.

Value

A matrix of x- and y-coordinates of the generated points. One row corresponds to one point.

Author(s)

Wei Zhang

Examples

```
numPoints <- 10
lowerObsCoords <- matrix(c(0, 0, 1, 0, 0, 1, 1, 1), nrow = 4, byrow = TRUE)
upperObsCoords <- matrix(c(1, 1, 2, 1, 1, 2, 2, 2), nrow = 4, byrow = TRUE)
s <- c(1, 1)
windowIntensities <- c(1:4)
lambda <- 0.1
stratRejectionSampler_exp(numPoints, lowerObsCoords, upperObsCoords, s, windowIntensities, lambda)
```

stratRejectionSampler_normal

Stratified rejection sampler for multivariate normal point process

Description

Simulate data using a stratified rejection sampler from a point process with an isotropic multivariate normal decay kernel.

Usage

```
stratRejectionSampler_normal(
  numPoints,
  lowerCoords,
  upperCoords,
  s,
  windowIntensities,
  sd
)
```

Arguments

`numPoints`	Number of spatial points to generate.
`lowerCoords, upperCoords`	Matrices of lower and upper x- and y-coordinates of a set of detection windows. One row for each window.
`s`	Vector of x- and y-coordinates of of the isotropic multivariate normal distribution mean.
`windowIntensities`	Vector of integrated intensities over all detection windows.
`sd`	Standard deviation of the isotropic multivariate normal distribution.

Value

A matrix of x- and y-coordinates of the generated points. One row corresponds to one point.

Author(s)

Joseph D. Chipperfield and Wei Zhang

Examples

```
numPoints <- 10
lowerObsCoords <- matrix(c(0, 0, 1, 0, 0, 1, 1, 1), nrow = 4, byrow = TRUE)
upperObsCoords <- matrix(c(1, 1, 2, 1, 1, 2, 2, 2), nrow = 4, byrow = TRUE)
s <- c(1, 1)
windowIntensities <- c(1:4)
sd <- 0.1
stratRejectionSampler_normal(numPoints, lowerObsCoords, upperObsCoords, s, windowIntensities, sd)
```

Index