Package 'MetaLandSim'

August 4, 2019

Type Package

Title Landscape and Range Expansion Simulation

Version 1.0.6

Date 2019-08-04

Depends R (>= 2.10), tcltk, igraph

Imports e1071, fgui, grDevices, graphics, googleVis, maptools, rgeos, rgrass7, raster, spatstat, stats, sp, minpack.lm, zipfR, coda, knitr

Suggests rasterVis

Author Frederico Mestre, Fernando Canovas, Benjamin Risk, Ricardo Pita, Antonio Mira, Pedro Beja.

Maintainer Frederico Mestre <mestre.frederico@gmail.com>

Description Tools to generate random landscape graphs, evaluate species occurrence in dynamic landscapes, simulate future landscape occupation and evaluate range expansion when new empty patches are available (e.g. as a result of climate change).

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2019-08-04 14:30:02 UTC

R topics documented:

`MetaLandSim-package` *Landscape And Range Expansion Simulation*

Description

The package MetaLandSim is a simulation environment, allowing the generation of random landscapes, represented as graphs, the simulation of landscape dynamics, metapopulation dynamics and range expansion.
The package was developed as part of the Ph.D. thesis of Frederico Mestre (SFRH/BD/73768/2010), funded by European Social Funds and the Portuguese Foundation for Science and Technology, and included in the project NETPERSIST (PTDC/AAG-MAA/3227/2012), funded by European Regional Development Fund (ERDF) through COMPETE programme and Portuguese national funds through the Portuguese Foundation for Science and Technology.
MetaLandSim is intended to provide a virtual environment, enabling the experimentation and simulation of processes at two scales: landscape and range. The simulation approach, taken by MetaLandSim, presents several advantages, like allowing the test of several alternatives and the knowledge of the full system (Peck, 2004; Zurell et al. 2009). The role of simulation in landscape ecology is fundamental due to the spatial and temporal scale of the studied phenomena, which frequently hinders experimentation (Ims, 2005).
Here, graph and metapopulation theories are combined, which is a broadly accepted strategy to provide a modelling framework for metapopulation dynamics (Cantwell & Forman, 1993; Bunn et al. 2000; Ricotta et al. 2000; Minor & Urban, 2008; Galpern et al. 2011). Also, several graph-based connectivity metrics can be computed from the landscape graphs. This set of metrics have been proven useful elsewhere (Urban & Keitt, 2001; Calabrese & Fagan, 2004). The graph representation of landscape has one major advantage: it effectively summarizes spatial relationships between elements and facilitates a multi-scale analysis integrating patch and landscape level analysis (Calabrese & Fagan, 2004).
MLS operates at two scales, providing researchers with the possibility of:

- Landscape scale - Simulation of metapopulation occupation on a dynamic landscape, computation of connectivity metrics.
- Range scale - Computes dispersal model and range expansion scenario simulation.

The landscape unit, an object of class `landscape`, is the basic simulation unit at both these scales. At the landscape scale, the persistence of the metapopulation in a dynamic landscape is evaluated through the simulation of landscape dynamics using the function `iterate.graph` or `manage_landscape_sim`. At the range scale the metapopulation is allowed to expand to other, empty, landscape units using `range_expansion`, producing an object of class `expansion`. The function `range_raster` allows the conversion of the dispersal model obtained with the previous function into a raster. Finally, also at the range scale, the user can analyse the outcome of several alternative landscapes in range expansion speed and maximum dispersal distance, using the function `manage_expansion_sim`.

Since version 1.0 new IFM parameter estimation capabilities are available, which based upon Bayesian statistics, using the functions first developed for the paper Risk et al.(2011).

We thank Dr. Santiago Saura (Universidad Politecnica de Madrid) for the very useful inputs and for the R script which greatly improved the connectivity metrics capabilities of MetaLandSim.

Reference paper: Mestre, F.; Canovas, F.; Pita, R.; Mira, A.; Beja. P. (2016). An R package for simulating metapopulation dynamics and range expansion under environmental change. Environmental Moldelling and Software, 81: 40-44.

Details

Package:	MetaLandSim
Type:	Package
Version:	1.0.5
Date:	2018-11-04
License:	GPL (>=2)

Author(s)

Frederico Mestre, Fernando Canovas, Benjamin Risk, Ricardo Pita, Antonio Mira and Pedro Beja.

Maintainer: Frederico Mestre <mestre.frederico@gmail.com>

References

Bunn, A. G., Urban, D. L. and Keitt, T. H. (2000). Landscape connectivity: a conservation application of graph theory. Journal of Environmental Management, 59(4), 265-278.

Calabrese, J. M. and Fagan, W. F. (2004). A comparison-shopper's guide to connectivity metrics. Frontiers in Ecology and the Environment, 2(10), 529-536.

Cantwell, M. D. and Forman, R. T. (1993). Landscape graphs: ecological modelling with graph theory to detect configurations common to diverse landscapes. Landscape Ecology, 8(4), 239-255.

Galpern, P., Manseau, M. and Fall, A. (2011). Patch-based graphs of landscape connectivity: a guide to construction, analysis and application for conservation. Biological Conservation, 144(1), 44-55.

Ims, R.A. (2005). The role of experiments in landscape ecology. In: Wiens, J.A., and Moss, M.R. (eds.). Issues and Perspectives in Landscape Ecology. Cambridge University Press. pp. 70-78.

Mestre, F., Pita, R., Pauperio, J., Martins, F. M., Alves, P. C., Mira, A., & Beja, P. (2015). Combining distribution modelling and non-invasive genetics to improve range shift forecasting. Ecological Modelling, 297, 171-179.

Mestre, F., Risk, B. B., Mira, A., Beja, P., & Pita, R. (2017). A metapopulation approach to predict species range shifts under different climate change and landscape connectivity scenarios. Ecological Modelling, 359, 406-414.

Minor, E. S. and Urban, D. L. (2008). A Graph Theory Framework for Evaluating Landscape Connectivity and Conservation Planning. Conservation Biology, 22(2), 297-307.

Peck, S. L. (2004). Simulation as experiment: a philosophical reassessment for biological modelling. Trends in Ecology & Evolution, 19(10), 530-534.

Ricotta, C., Stanisci, A., Avena, G. C., and Blasi, C. (2000). Quantifying the network connectivity of landscape mosaics: a graph-theoretical approach. Community Ecology, 1(1), 89-94.

Risk, B. B., De Valpine, P., Beissinger, S. R. (2011). A robust design formulation of the incidence function model of metapopulation dynamics applied to two species of rails. Ecology, 92(2), 462-474.

Urban, D. and Keitt, T. (2001). Landscape connectivity: a graph-theoretic perspective. Ecology, 82(5), 1205-1218.

Zurell, D., Berger, U., Cabral, J.S., Jeltsch, F., Meynard, C.N., Munkemuller, T., Nehrbass, N., Pagel, J., Reineking, B., Schroder, B. and Grimm, V. (2009). The virtual ecologist approach: simulating data and observers. Oikos, 119(4), 622-635.

`accept.calculate` *Calculate acceptance rates in MCMC chains*

Description

Calculate acceptance rates of parameters in the IFM.

Usage

```
accept.calculate(x, model = c("naive", "missing", "robust"))
```

Arguments

`x`	A named list with the MCMC chains estimated by ifm.naive.MCMC, ifm.missing.MCMC, or ifm.robust.MCMC.
`model`	Either "naive", "missing", or "robust"

Value

Named list containing MCMC chain acceptance rates. Names are built from the input list, e.g., for model="naive":

`acc.b.chain`	Acceptance rates of parameter b
`acc.e.chain`	Acceptance rates of parameter e
`acc.y.chain`	Acceptance rates of parameter y
`acc.alpha.chain`	Acceptance rates of parameter alpha
`acc.x.chain`	Acceptance rates of parameter x

Author(s)

Benjamin Risk

Examples

```
data(simulatedifm)

# Here, we run a chain with random initial values:
init1=list(alpha=runif(1,1,30), b=runif(1,0,5),y=runif(1,0,20),e=runif(1,0,1),x=runif(1,0,5))

inm1 <- ifm.naive.MCMC(niter=1000,init=init1,z.data =
 z.sim,site.distance=sim.distance,site.area=sim.area,
  sd.prop.alpha=4,sd.prop.b=0.6,sd.prop.y=40,sd.prop.e=0.05,sd.prop.x=0.4,nthin=1,print.by=100)
accept.calculate(inm1,model='naive')
```

---

addpoints *Add a given number of patches to a landscape*

Description

Adds a given number of patches to the landscape.

Usage

```
addpoints(rl, nr)
```

Arguments

rl	Object of class 'landscape'.
nr	Number of patches to be added (see 'note').

Value

Returns an object of class 'landscape'.

Note

The number of patches to be added might be impaired by the minimum distance between points.

Author(s)

Frederico Mestre and Fernando Canovas

See Also

rland.graph, removepoints

Examples

```
data(rland)

#Checking the number of patches in the starting landscape:

rland$number.patches

#60

#Adding 10 patches to a landscape:

rl1 <- addpoints(rl=rland, nr=10)

#Checking the number of patches in the output landscape:

rl1$number.patches

#70
```

---

cabrera *Modified patch occupancy data of Cabrera vole*

Description

One season patch occupancy dataset for *Microtus cabrerae* in SW Portugal. This dataset is in the format produced by species.graph, convert.graph or import.shape (class 'metapopulation'), and it was created by converting a data frame using the function convert.graph. The data frame had the information of one snapshot of patch occupancy data of Cabrera vole (Microtus cabrera) in southwestern Portugal.

Usage

```
data(cabrera)
```

Format

A list with the following elements:

- mapsize - 8200 (landscape mosaic side length, in meters).
- minimum.distance - 10.04 (minimum distance between patches centroids).
- mean.area - 0.46 (mean area, in hectares).
- SD.area - 1.05 (SD of the area).
- number.patches - 793 (number of patches).
- dispersal - 800 (mean dispersal ability of the species).
- distance.to.neighbours - data frame with pairwise distance between patches.
- nodes.characteristics - data frame with the characteristics of each patch.

Details

To create this sample dataset the occupancy status of patches was scrambled, however the proportion of occupied patches was kept.

Source

Original field data was obtained during project PERSIST (PTDC/BIA-BEC/105110/2008).

Examples

```
data(cabrera)
```

`calcmode` *Function for mode estimation of a continuous variable*

Description

Derives the mode, estimating the value of a continuous variable.

Usage

```
calcmode(data,adjust=1)
```

Arguments

`data`	vector used to estimate the mode.
`adjust`	increase this value to make the density estimate smoother.

Value

Returns the numeric value of the mode.

Author(s)

Adapted from https://stat.ethz.ch/pipermail/r-help/2008-August/172323.html.

Examples

```
vect1 = rchisq(1000,df=3)
calcmode(vect1)
vect1
```

`cluster.graph` *Delivers the number of patches per cluster*

Description

Returns a data frame with the number of nodes (habitat patches) in each component of the landscape graph (in this case a component is a group of patches connected by the species dispersal distance).

Usage

```
cluster.graph(rl)
```

Arguments

`rl`	Object of class 'landscape'.

Details

The components are defined based on the species mean dispersal ability. This implies that the connectivity model between patches is binary (connected/not connected) as opposed to probabilistic.

Value

This function returns a data frame with the number of patches of each component (group of patches). The returned data frame has two fields: cluster (Id of the component) and number of nodes (the number of nodes of the respective component).

Author(s)

Frederico Mestre and Fernando Canovas

See Also

`rland.graph`

Examples

```
data(rland)

cluster.graph(rl=rland)

#Output:

#  cluster number of nodes
#1       1              11
#2       2               1
#3       3              13
#4       4               1
```

```
#5        5              1
#6        6             15
#7        7              2
#8        8              1
#9        9              3
#10      10              1
#11      11              1
#12      12              2
#13      13              4
#14      14              1
#15      15              1
#16      16              1
#17      17              1
```

`cluster.id` *Classify patches in clusters*

Description

reclassify clusters of a landscape according to a given mean dispersal distance.

Usage

```
cluster.id(rl)
```

Arguments

`rl`	Object of class 'landscape'.

Details

After changing the landscape some components (groups of connected patches) might suffer changes (e.g. the removal of patches might split components). This function re-attributes a code to each patch, identifying the groups of connected patches (components), after this type of disturbance to the habitat network.Mainly to be used internally.

Value

Returns the same landscape object, with the clusters reclassified.

Author(s)

Frederico Mestre and Fernando Canovas

See Also

`rland.graph`

Examples

```
data(rland)

#After removing 30 (50%) of the patches of a landscape:

rland2 <- removepoints(rl=rland, nr=35)

#A reclassification might be needed to identify components:

rland2 <- cluster.id(rl=rland2)

#After removing 35 patches, there's a different number of components:

components.graph(rl=rland)

#21

components.graph(rl=rland2)

#16
```

`coda.create` *Create files for use with R-package coda.*

Description

Creates two text files, <filename.txt> and filename_Index.txt>, in the format used by OpenBUGS, which can then be read using read.coda() to create an mcmc object for subsequent use of coda diagnostic and plotting functions.

Usage

```
coda.create(object, file.name, write.index = TRUE, par.list = list("mupsi1.chain",
"e.chain", "x.chain", "b.chain", "y.chain", "alpha.chain"), niter = 101000, nthin = 10)
```

Arguments

`object`	Output from ifm.naive.MCMC, ifm.missing.MCMC, or ifm.robust.MCMC, i.e., a named list with vectors e.chain. x.chain, y.chain, b.chain, alpha.chain, and other parameters depending on the model.
`file.name`	Name of the text file that will be output in the format used by BUGS and JAGS.
`write.index`	Logical indicating whether or not to also create file.name_Index.txt. Defaults to TRUE, which allows the subsequent use of read.coda().
`par.list`	List of parameters to include in the file. Defaults to mupsi1.chain, e.chain, x.chain, b.chain, y.chain, alpha.chain; modify to include additional parameters in ifm.missing.MCMC and ifm.robust.MCMC.

niter	Number of iterations in the original chain (before thinning).
nthin	Thinning used in the original estimation function; this is just bookkeeping, as this function does not perform the thinning.

Details

Writes text files to the current working directory or to the path specified with "file.name".

Author(s)

Benjamin Risk

Examples

```
## Not run:

# quick run; actual estimation requires more iterations:

data(simulatedifm)

myniter=100
nsite=nrow(z.sim)
nyear=ncol(z.sim)
nthin=1
nburnin=0
init1=list(alpha=runif(1,1,30), b=runif(1,0,5),y=runif(1,0,20),e=runif(1,0,1),x=runif(1,0,5))

inm1 <- ifm.naive.MCMC(niter=myniter,init=init1,z.data =
 z.sim,site.distance=sim.distance,site.area=sim.area,
  sd.prop.alpha=4,sd.prop.b=0.6,sd.prop.y=40,sd.prop.e=0.05,sd.prop.x=0.4,nthin=1,print.by=1000)

# write files in OpenBUGS format to working directory:
coda.create(inm1,"sim_inm1",par.list=list("e.chain","x.chain","alpha.chain",
"b.chain","y.chain"),niter=myniter,nthin=nthin)

## End(Not run)
```

---

combine.chains *Combines two chains into a single chain.*

Description

Combines two lists of chains from ifm.naive.MCMC, ifm.missing.MCMC, or ifm.robust.MCMC into one list where each element is the concatenated chains.

Usage

```
combine.chains(x1, x2, nburnin, nthin = 1, z.thin = TRUE)
```

Arguments

x1	First list of chains.
x2	Second list of chains.
nburnin	Number of initial iterations to discard.
nthin	If nthin>1, subsets to every nthin^th sample
z.thin	logical; defaults to TRUE. Thinning for the posterior sample of the occupancy states. If true, uses thinning equal to 5. The posterior sample of occupancy states is a large nsite x nyear x niter array, and this option reduces memory usage. Ignored if the chain is from the ifm.naive.MCMC (where occupancy states are fixed).

Value

Named list with the same names as the inputs x1 and x2

Author(s)

Benjamin Risk

Examples

```
data(simulatedifm)

init1=list(alpha=runif(1,1,30), b=runif(1,0,5),y=runif(1,0,20),e=runif(1,0,1),x=runif(1,0,5))

inm1 <- ifm.naive.MCMC(niter=500,init=init1,z.data =
 z.sim,site.distance=sim.distance,site.area=sim.area,
  sd.prop.alpha=4,sd.prop.b=0.6,sd.prop.y=40,sd.prop.e=0.05,sd.prop.x=0.4,nthin=1,print.by=100)

init1=list(alpha=runif(1,1,30), b=runif(1,0,5),y=runif(1,0,20),e=runif(1,0,1),x=runif(1,0,5))

inm2 <- ifm.naive.MCMC(niter=500,init=init1,z.data =
 z.sim,site.distance=sim.distance,site.area=sim.area,
  sd.prop.alpha=4,sd.prop.b=0.6,sd.prop.y=40,sd.prop.e=0.05,sd.prop.x=0.4,nthin=1,print.by=100)

sim.inm=combine.chains(inm1,inm2,nburnin=0,nthin=1)
```

---

components.graph *Number of components of a landscape*

Description

Returns the number of components in the landscape graph (in this case a component is a group of patches connected by the species dispersal distance).

Usage

```
components.graph(rl)
```

Arguments

rl	Object of class 'landscape'.

Value

Returns the number of components (groups of connected patches) of a landscape.

Author(s)

Frederico Mestre and Fernando Canovas

See Also

rland.graph

Examples

```
data(rland)

components.graph(rl=rland)

#21
```

---

convert.graph *Convert data frame to landscape*

Description

Converts a given data frame in a list which can be used in the following functions, an object of class 'metapopulation'.

Usage

```
convert.graph(dframe, mapsize, dispersal)
```

Arguments

dframe	data frame with the original data and the following columns, in this order: • ID - patch Id. • X - Coordinate. • Y - Coordinate. • Area - Patch area, in hectares. • Occupation - Species presence status (0/1).
mapsize	Landscape mosaic side length, in meters.
dispersal	Species mean dispersal ability, in meters.

Value

Delivers an object of class 'metapopulation'.

Author(s)

Frederico Mestre and Fernando Canovas

See Also

species.graph

Examples

```
data(mc_df)

#Checking the columns of the data frame:

head(mc_df)

#  ID        x       y  area mc
#1  1 1248.254   0.000 0.079  0
#2  2 1420.857  46.725 0.781  1
#3  3 1278.912  52.629 1.053  1
#4  4 6370.625  62.637 0.788  0
#5  5 1151.337  97.140 0.079  0
#6  6 1295.796 104.839 0.137  1


#In order to import the data frame mc_df:

sp1 <- convert.graph(dframe=mc_df, mapsize=8300, dispersal=800)

#verify class

class(sp1)

# [1] "metapopulation"
```

---

create.parameter.df *Create parameter data frame*

Description

This function creates a parameter data frame, using parameter values computed with the application available in the papers of Moilanen (1999) and ter Braak and Etienne (2003).

Usage

```
create.parameter.df(alpha, x, y, e)
```

Arguments

alpha	Alpha parameter
x	x parameter
y	y parameter
e	e parameter

Details

It is highly recommended that the user reads both papers, as well as the help files.

Value

Returns a data frame, with the same format as the one returned by parameter.estimate for the methods 'Rsnap_1' and 'Rsnap_x'.

Author(s)

Frederico Mestre and Fernando Canovas

References

Moilanen, A. (1999). Patch occupancy models of metapopulation dynamics: efficient parameter estimation using implicit statistical inference. Ecology, 80(3): 1031-1043.

ter Braak, C. J., & Etienne, R. S. (2003). Improved Bayesian analysis of metapopulation data with an application to a tree frog metapopulation. Ecology, 84(1): 231-241.

See Also

parameter.estimate

Examples

```
param2 <- create.parameter.df(alpha=0.5, x=0.1, y=5, e=0.1)

param2

#      par_output
#alpha        0.5
#x            0.1
#y            5.0
#e            0.1
```

edge.graph *Produce an edge (links) data frame*

Description

Returns a data frame with the information on the connections between patches (assuming binary connections).

Usage

```
edge.graph(rl)
```

Arguments

rl	Object of class 'landscape'.

Value

Produces a data frame with the information on the edges (links): the IDs of both patches, the area, the coordinates and the Euclidean distance.

Author(s)

Frederico Mestre and Fernando Canovas

See Also

rland.graph

Examples

```
data(rland)

edge_df <- edge.graph(rl=rland)
```

expansion *Class 'expansion'*

Description

Class representing an expansion object, as produced by range_expansion.

Slots

A list of four data frames with the proportion of occupation at several distances from the closest occupied landscape mosaic. These four data frames correspond to the proportion of occupation to the north, south, east and west. Each data frame has the following columns:

- DISTANCE - Distance (mapsize x number of landscapes).
- OCCUPATION - How many times did the landscape at this distance got occupied by the species (from a total of 'iter' repetitions).
- PROPORTION - Proportion of occupation for the landscape at this distance (OCCUPATION/iter).

Author(s)

Frederico Mestre and Fernando Canovas

extract.graph *Extract landscape from span.graph generated list*

Description

Extracts a landscape from an object delivered by span.graph. The output is an object of class 'landscape'.

Usage

```
extract.graph(rl, rlist, nr)
```

Arguments

rl	Object of class 'landscape' used to generate the list, with span.graph.
rlist	Object delivered by span.graph.
nr	Position of the landscape in the list (rlist).

Value

Delivers an object of class 'landscape'.

Author(s)

Frederico Mestre and Fernando Canovas

See Also

span.graph, rland.graph

Examples

```
data(rland)
data(landscape_change)

#Extracting the landscape of the 50th time step:

rl1 <- extract.graph(rl=rland, rlist=landscape_change, nr=50)
```

---

ifm.missing.MCMC *Estimate the 'missing' design incidence function model*

Description

Estimates the IFM with no false absences but incorporating missing data.

Usage

```
ifm.missing.MCMC(niter=1000,init,z.data, site.distance, site.area,
sd.prop.mupsi1=0.1, sd.prop.e=0.1, sd.prop.x=0.5,sd.prop.y=10, sd.prop.b=0.2,
sd.prop.alpha=5, nthin=1,nsite.subset=10,print.by=100)
```

Arguments

niter	Number of iterations in the MCMC chain.
init	Named list with values to initialize the chain. E.g.:
	init1=list(z.missing=runif(nmissing),mupsi1=runif(1),alpha=runif(1,1,30), b=runif(1,0,5),y=runif(1,0,20),
	e=runif(1,0,1),x=runif(1,0,5)).
	z.missing: a vector of initial occupancy states for the missing data with length equal to the number of NAs in z.data (i.e., vectorized across years). Can use runif(nmissing).
	mupsi1: probability of initial occupancy in year 1; runif(1) suffices

alpha: initial value for alpha in dispersal model; described as 1 / average dispersal distance

b: initial value for parameter b in colonization model

y: initial value for parameter y in colonization model

e: initial value for e in extinction model

x: initial value for x in extinction model

`z.data` nsite x nyears matrix containing NA for missing data. Occupancy at sites with missing data will be estimated.

`site.distance` nsite x nsite matrix of distances between sites. The tuning parameters in the example are set for distances less than one, with max distance approximately 0.5. Input data should have a similar scaling.

`site.area` Vector of length nsite with areas. The tuning parameters in the example are set for average area approximately equal to 1. Input data should have a similar scaling.

`sd.prop.mupsi1` Standard deviation of the proposal distribution for occupancy in year 1.

`sd.prop.e` Standard deviation of the proposal distribution for parameter e.

`sd.prop.x` Standard deviation of the proposal distribution for parameter x.

`sd.prop.y` Standard deviation of the proposal distribution for parameter y.

`sd.prop.b` Standard deviation of the proposal distribution for parameter b.

`sd.prop.alpha` Standard deviation of the proposal distribution for parameter alpha.

`nthin` If specified, keeps only every nthin^th sample from the MCMC chain. Use to save memory or when the chain is moving slowly.

`nsite.subset` The number of sites to include in the block sampling, where nsite.subset is equal to the number of sites updated in the same step. Larger values decrease the probability of acceptance.

`print.by` Specifies how often to print the number of the current iteration.

Value

`z.chain` nsite x nyear x niter array sampled from the posterior distribution of occupancy in each year (if detection occurred at a given year and site, then the value is identically equal to one for all iterations).

`muz.chain` nyear x niter matrix posterior sample of the proportion of sites occupied in each year.

`muz.missing.chain`
nyear x niter matrix posterior sample of the proportion of sites occupied for sites with missing data.

`prop.extinct.chain`
Extinction rate for all sites.

`prop.colon.chain`
Colonization rate.

mupsi1.chain	posterior sample of parameter for occupancy in year 1.
e.chain	posterior sample of e
x.chain	posterior sampmle of x
y.chain	posterior sample of y
b.chain	posterior sample of b
alpha.chain	posterior sample of alpha

Author(s)

Benjamin Risk

References

Risk, B. B., De Valpine, P., Beissinger, S. R. (2011). A robust design formulation of the incidence function model of metapopulation dynamics applied to two species of rails. Ecology, 92(2), 462-474.

Examples

```
## Not run:

data(simulatedifm)
library("coda")

niter=2000
nsite=100
nyear=10
nthin=1
nburnin=1000
## NOTE! The notation used here corresponds to MetaLandSim and differs from Risk et al 2011
## Here
## e (in MetaLandSim) = mu (in Risk et al 2011)
## x = chi
## y = gamma
## b = beta
## alpha = alpha
##
# Priors:
#         e: [0,1]
#         x: [0,5]
#         y^2: [0,400]
#         b: [0,5]
#         alpha: [1,30]

# NOTE: If posteriors are truncated at zero, then estimates may be biased. Rescale
# distances (e.g., divide by 10,000) and/or areas so that parameters are larger.

nmissing = sum(is.na(z.sim.20))

init1=list(z.missing=runif(nmissing),mupsi1=runif(1),alpha=runif(1,1,30),
```

```
 b=runif(1,0,5),y=runif(1,0,20),e=runif(1,0,1),x=runif(1,0,5))

a = Sys.time()
im1 <- ifm.missing.MCMC(niter=niter,init=init1,z.data = z.sim.20,
 site.distance=sim.distance,site.area=sim.area, sd.prop.mupsi1=0.2, sd.prop.alpha=4, sd.prop.b=0.6,
 sd.prop.y=40, sd.prop.e=0.05, sd.prop.x=0.4, nthin=1, print.by=500)
accept.calculate(im1,model='missing')
Sys.time() - a


init2=list(z.missing = runif(nmissing), mupsi1 = runif(1), alpha=runif(1,1,30),
 b=runif(1,0,5),y=runif(1,0,20),e=runif(1,0,1),x=runif(1,0,5))
im2 <- ifm.missing.MCMC(niter=niter,init=init2, z.data = z.sim.20, site.distance=sim.distance,
site.area=sim.area, sd.prop.mupsi1=0.2, sd.prop.alpha=4, sd.prop.b=0.6, sd.prop.y=40,
sd.prop.e=0.05,sd.prop.x=0.4, nthin=1, print.by=1000)
accept.calculate(im2,model='missing')
Sys.time() - a

coda.create(im1,"sim_im1",par.list=list("mupsi1.chain","e.chain","x.chain","alpha.chain",
"b.chain","y.chain"),niter=niter,nthin=nthin)
coda.create(im2,"sim_im2",par.list=list("mupsi1.chain","e.chain","x.chain","alpha.chain",
"b.chain","y.chain"),niter=niter,nthin=nthin)
coda.sim.im1=read.coda("sim_im1.txt","sim_im1_Index.txt")
coda.sim.im2=read.coda("sim_im2.txt","sim_im2_Index.txt")
coda.sim.im.list=mcmc.list(coda.sim.im1,coda.sim.im2)
sim.im=combine.chains(im1,im2,nburnin=nburnin,nthin=1)
coda.create(sim.im,"sim_im",par.list=list("mupsi1.chain","e.chain","x.chain","alpha.chain",
"b.chain","y.chain"),niter=(2*niter-2*nburnin),nthin=nthin)
coda.sim.im.long=read.coda("sim_im.txt","sim_im_Index.txt")

summary(coda.sim.im.list)
summary(coda.sim.im.long)

gelman.diag(coda.sim.im.list)

plot(coda.sim.im.list)
plot(coda.sim.im.long)
cumuplot(coda.sim.im.long)

# calculate maximum a posteriori estimates:
m1 <- as.matrix(sim.im)
e <- calcmode(m1[,1][[1]])
x <- calcmode(m1[,1][[2]])
y <- calcmode(m1[,1][[3]])
b <- calcmode(m1[,1][[4]])
alpha <- calcmode(m1[,1][[5]])


## End(Not run)
```

`ifm.naive.MCMC` *Estimate the naive design incidence function model*

Description

Estimates the IFM assuming no false absences and omitting sites for particular years in which data were missing.

Usage

```
ifm.naive.MCMC(niter=1000,init,z.data, site.distance, site.area, sd.prop.e=0.2,
 sd.prop.x=0.5,sd.prop.y=10, sd.prop.b=0.2, sd.prop.alpha=5,nthin=1,print.by=100)
```

Arguments

`niter`	Number of iterations in the MCMC chain.
`init`	Named list with values to initialize the chain. E.g.: init1=list(alpha=runif(1,1,30), b=runif(1,0,5),y=runif(1,0,20), e=runif(1,0,1),x=runif(1,0,5)). alpha: initial value for alpha in dispersal model; described as 1 / average dispersal distance b: initial value for parameter b in colonization model y: initial value for parameter y in colonization model e: initial value for e in extinction model x: initial value for x in extinction model
`z.data`	nsite x nyears matrix. If contains NAs, the corresponding parts are omitted from the likelihood (the missing data are not estimated).
`site.distance`	nsite x nsite matrix of distances between sites. The tuning parameters in the example are set for distances less than one, with max distance approximately 0.5.
`site.area`	Vector of length nsite with areas. The tuning parameters in the example are set for average area approximately equal to 1.
`sd.prop.e`	Standard deviation of the proposal distribution for parameter e.
`sd.prop.x`	Standard deviation of the proposal distribution for parameter x.
`sd.prop.y`	Standard deviation of the proposal distribution for parameter y.
`sd.prop.b`	Standard deviation of the proposal distribution for parameter b.
`sd.prop.alpha`	Standard deviation of the proposal distribution for parameter alpha.
`nthin`	If specified, keeps only every nthin^th sample from the MCMC chain. Use to save memory or when the chain is moving slowly.
`print.by`	Specifies how often to print the number of the current iteration.

Value

e.chain	posterior sample of e
x.chain	posterior sampmle of x
y.chain	posterior sample of y
b.chain	posterior sample of b
alpha.chain	posterior sample of alpha
deviance.chain	posterior sample of -2*loglik

Author(s)

Benjamin Risk

References

Risk, B. B., De Valpine, P., Beissinger, S. R. (2011). A robust design formulation of the incidence function model of metapopulation dynamics applied to two species of rails. Ecology, 92(2), 462-474.

Examples

```
## Not run:
data(simulatedifm)

library("coda")

myniter=5000
nsite=nrow(z.sim)
nyear=ncol(z.sim)
nthin=1
nburnin=1000
## NOTE! The notation used here corresponds to MetaLandSim and differs from Risk et al 2011
## Here
## e (in MetaLandSim) = mu
## x = chi
## y = gamma
## b = beta
## alpha = alpha
##
# Priors:
#         e: [0,1]
#         x: [0,5]
#         y^2: [0,400]
#         b: [0,5]
#         alpha: [1,30]

# NOTE: If posteriors are truncated at zero, then estimates are biased. Rescale
# distances (e.g., divide by 10,000) and/or areas so that parameters are larger.

# Here, we run two chains with random initial values:
init1=list(alpha=runif(1,1,30), b=runif(1,0,5),y=runif(1,0,20),e=runif(1,0,1),x=runif(1,0,5))
```

```
a = Sys.time()
inm1 <- ifm.naive.MCMC(niter=myniter,init=init1,z.data =
 z.sim,site.distance=sim.distance,site.area=sim.area,
  sd.prop.alpha=4,sd.prop.b=0.6,sd.prop.y=40,sd.prop.e=0.05,sd.prop.x=0.4,nthin=1,print.by=1000)
accept.calculate(inm1,model='naive')
Sys.time() - a

init2=list(alpha=runif(1,1,30), b=runif(1,0,5),y=runif(1,0,20),e=runif(1,0,1),x=runif(1,0,5))
inm2 <- ifm.naive.MCMC(niter=myniter,init=init2,z.data =
z.sim,site.distance=sim.distance,site.area=sim.area,
sd.prop.alpha=4,sd.prop.b=0.6,sd.prop.y=40,sd.prop.e=0.05,sd.prop.x=0.4,nthin=1,print.by=1000)
accept.calculate(inm2,model='naive')
Sys.time() - a

coda.create(inm1,"sim_inm1",par.list=list("e.chain","x.chain","alpha.chain",
"b.chain","y.chain"),niter=myniter,nthin=nthin)
coda.create(inm2,"sim_inm2",par.list=list("e.chain","x.chain","alpha.chain",
"b.chain","y.chain"),niter=myniter,nthin=nthin)
coda.sim.inm1=read.coda("sim_inm1.txt","sim_inm1_Index.txt")
coda.sim.inm2=read.coda("sim_inm2.txt","sim_inm2_Index.txt")
coda.sim.inm.list=mcmc.list(coda.sim.inm1,coda.sim.inm2)
sim.inm=combine.chains(inm1,inm2,nburnin=nburnin,nthin=1)
coda.create(sim.inm,"sim_inm",par.list=list("e.chain","x.chain","alpha.chain",
"b.chain","y.chain"),niter=(2*myniter-2*nburnin),nthin=nthin)
coda.sim.inm.long=read.coda("sim_inm.txt","sim_inm_Index.txt")

summary(coda.sim.inm.list)
summary(coda.sim.inm.long)

gelman.diag(coda.sim.inm.list)

plot(coda.sim.inm.list)
plot(coda.sim.inm.long)
cumuplot(coda.sim.inm.long)

# calculate maximum a posteriori estimates:
m1 <- as.matrix(sim.inm)
e <- calcmode(m1[,1][[1]])
x <- calcmode(m1[,1][[2]])
y <- calcmode(m1[,1][[3]])
b <- calcmode(m1[,1][[4]])
alpha <- calcmode(m1[,1][[5]])


## End(Not run)
```

`ifm.robust.MCMC` *Estimate the robust design incidence function model*

Description

Estimates the IFM with imperfect detection and missing data.

Usage

```
ifm.robust.MCMC(niter = 1000, init, det.data, site.distance, site.area, sd.prop.p = 0.1,
sd.prop.mupsi1 = 0.1, sd.prop.e = 0.2, sd.prop.x = 0.2, sd.prop.y = 0.2, sd.prop.b = 0.2,
sd.prop.alpha = 0.2, nthin = 1, nsite.subset = 5, print.by = 100)
```

Arguments

niter	Number of iterations in the MCMC chain.
init	Named list with values to initialize the chain. E.g.: init1=list(z.data=initocc,z.missing=runif(nmissing),p=runif(nyear, 0.1,1),mupsi1=runif(1),alpha=runif(1,1,30), b=runif(1,0,5),y=runif(1,0,20), e=runif(1,0,1),x=runif(1,0,5)). z.data: a matrix with nrows = number of sites and ncol = number of years. Contains NAs for missing values. Contains naive estimates of occupancy elsewhere. z.missing: z.missing: a vector of initial occupancy states for the missing data with length equal to the number of NAs in z.data (i.e., vectorized across years). Can use runif(nmissing). p: vector of length nyears with inital probability of detection in each year mupsi1: probability of initial occupancy in year 1; runif(1) suffices alpha: initial value for alpha in dispersal model; described as 1 / average dispersal distance b: initial value for parameter b in colonization model y: initial value for parameter y in colonization model e: initial value for e in extinction model x: initial value for x in extinction model
det.data	Detection data in an array with dimensions nsites x nyears x nvisits. For removal design, set all values after a detection equal to NA. For missing data in a given year, set all visits to NA.
site.distance	nsite x nsite matrix of distances between sites. The tuning parameters in the example are set for distances less than one, with max distance approximately 0.5. Input data should have a similar scaling.

site.area	Vector of length nsite with areas. The tuning parameters in the example are set for average area approximately equal to 1. Input data should have a similar scaling.
sd.prop.p	Scalar equal to the standard deviation of the proposal distribution for probability of detection, which is a normal distribution centered at current value in the mcmc chain. The same standard deviation is used for all years.
sd.prop.mupsi1	Standard deviation of the proposal distribution for occupancy in year 1.
sd.prop.e	Standard deviation of the proposal distribution for parameter e.
sd.prop.x	Standard deviation of the proposal distribution for parameter x.
sd.prop.y	Standard deviation of the proposal distribution for parameter y.
sd.prop.b	Standard deviation of the proposal distribution for parameter b.
sd.prop.alpha	Standard deviation of the proposal distribution for parameter alpha.
nthin	If specified, keeps only every nthin^th sample from the MCMC chain. Use to save memory or when the chain is moving slowly.
nsite.subset	The number of sites to include in the block sampling, where nsite.subset is equal to the number of sites updated in the same step. Larger values decrease the probability of acceptance.
print.by	Specifies how often to print the number of the current iteration.

Value

z.chain	nsite x nyear x niter array sampled from the posterior distribution of occupancy in each year (if detection occurred at a given year and site, then the value is identically equal to one for all iterations).
muz.chain	nyear x niter matrix posterior sample of the proportion of sites occupied in each year.
muz.missing.chain	nyear x niter matrix posterior sample of the proportion of sites occupied for sites with missing data.
prop.extinct.chain	Extinction rate for all sites.
prop.colon.chain	Colonization rate.
p.chain	nyear x niter sample of detection probabilities.
mupsi1.chain	posterior sample of parameter for occupancy in year 1.
e.chain	posterior sample of e
x.chain	posterior sampmle of x
y.chain	posterior sample of y
b.chain	posterior sample of b
alpha.chain	posterior sample of alpha
latent.deviance.chain	posterior sample of -2*loglik

Author(s)

Benjamin Risk

References

Risk, B. B., De Valpine, P., Beissinger, S. R. (2011). A robust design formulation of the incidence function model of metapopulation dynamics applied to two species of rails. Ecology, 92(2), 462-474.

Examples

```
## Not run:

data(simulatedifm)
library("coda")

# There are more parameters in this model
# and estimating the posterior requires more iterations:
niter=2000
nsite=nrow(z.sim)
nyear=ncol(z.sim)
nthin=1
nburnin=1000
## NOTE! The notation used here corresponds to MetaLandSim and differs from Risk et al 2011
## Here
## e (in MetaLandSim) = mu (in Risk et al 2011)
## x = chi
## y = gamma
## b = beta
## alpha = alpha
##
# Priors:
#         e: [0,1]
#         x: [0,5]
#         y^2: [0,400]
#         b: [0,5]
#         alpha: [1,30]

# NOTE: If posteriors are truncated at zero, then estimates are biased. Rescale
# distances (e.g., divide by 10,000) and/or areas so that parameters are larger.

# Count number of times a site was never visited in a given year:
nmissing = sum(is.na(z.sim.20))

# Create a dataset with initial guess of true occupancy for sites with visits.
# This dataset should be number of sites by years
# one way of generating these initial values:
initocc <- suppressWarnings(apply(sim.det.20,c(1,2),max,na.rm=TRUE))
# produces warnings but that's okay
initocc[initocc==-Inf]=NA
```

```
init1=list(z.data=initocc,z.missing=runif(nmissing),p=runif(nyear,
0.1,1),mupsi1=runif(1),alpha=runif(1,1,30), b=runif(1,0,5),y=runif(1,0,20),e=runif(1,0,1)
,x=runif(1,0,5))

# for diagnosing acceptance rates:
# init1=list(z.data=initocc,z.missing=runif(nmissing),p=runif(nyear,0.1,1),
mupsi1=runif(1),alpha=20, b=0.5,y=7.5,e=0.25,x=0.25)

a = Sys.time()
ir1 <- ifm.robust.MCMC(niter=niter,init=init1, det.data = sim.det.20,
site.distance=sim.distance,site.area=sim.area, sd.prop.p=0.25,sd.prop.mupsi1=0.2,
sd.prop.alpha=2, sd.prop.b=0.6, sd.prop.y=20, sd.prop.e=0.1, sd.prop.x=0.4, nthin=1)
accept.calculate(ir1,model='robust')
Sys.time() - a

init2=list(z.data=initocc,z.missing=runif(nmissing),p=runif(nyear,
0.1,1),mupsi1=runif(1),alpha=runif(1,1,30), b=runif(1,0,5),y=runif(1,0,20),
e=runif(1,0,1),x=runif(1,0,5))
ir2 <- ifm.robust.MCMC(niter=niter,init=init2, det.data = sim.det.20,
 site.distance=sim.distance,site.area=sim.area, sd.prop.mupsi1=0.2, sd.prop.alpha=2, sd.prop.b=0.6,
   sd.prop.y=20, sd.prop.e=0.1, sd.prop.x=0.4, sd.prop.p = 0.25, nthin=1)
accept.calculate(ir2,model='robust')

coda.create(ir1,"sim_ir1",par.list=list("mupsi1.chain","e.chain","x.chain",
"alpha.chain","b.chain","y.chain","p.chain"),niter=niter,nthin=nthin)
coda.create(ir2,"sim_ir2",par.list=list("mupsi1.chain","e.chain","x.chain",
"alpha.chain","b.chain","y.chain","p.chain"),niter=niter,nthin=nthin)
coda.sim.ir1=read.coda("sim_ir1.txt","sim_ir1_Index.txt")
coda.sim.ir2=read.coda("sim_ir2.txt","sim_ir2_Index.txt")
coda.sim.ir.list=mcmc.list(coda.sim.ir1,coda.sim.ir2)
sim.ir=combine.chains(ir1,ir2,nburnin=nburnin,nthin=1)
coda.create(sim.ir,"sim_ir",par.list=list("mupsi1.chain","e.chain",
"x.chain","alpha.chain","b.chain","y.chain","p.chain"),niter=(2*niter-2*nburnin),nthin=nthin)
coda.sim.ir.long=read.coda("sim_ir.txt","sim_ir_Index.txt")

summary(coda.sim.ir.list)
summary(coda.sim.ir.long)

gelman.diag(coda.sim.ir.list)

plot(coda.sim.ir.list)
plot(coda.sim.ir.long)
cumuplot(coda.sim.ir.long)

# calculate maximum a posteriori estimates:
m1 <- as.matrix(sim.ir)
e <- calcmode(m1[,1][[1]])
x <- calcmode(m1[,1][[2]])
y <- calcmode(m1[,1][[3]])
b <- calcmode(m1[,1][[4]])
alpha <- calcmode(m1[,1][[5]])
```

```
## End(Not run)
```

`import.shape` *Import a shapefile*

Description

Imports a shapefile, converting it to an object of class 'metapopulation'.

Usage

```
import.shape(filename, path, species.col, ID.col, area.col, dispersal)
```

Arguments

`filename`	Character vector with the shapefile name.
`path`	Character vector with the path to the file.
`species.col`	Character vector with the name of the column (in the shapefile) with the species occupancy data.
`ID.col`	Character vector with the name of the column (in the shapefile) with the patch Id.
`area.col`	Character vector with the name of the column (in the shapefile) with the patch area, in hectares.
`dispersal`	Species mean dispersal ability, in meters.

Value

Delivers an object of class 'metapopulation'.

Note

The shapefile must be in project coordinates (units=meters and hectares).

Author(s)

Frederico Mestre and Fernando Canovas

See Also

`rland.graph`, `convert.graph`

Examples

```
## Not run:

rl1 <- import.shape(filename = "yourshapefile.shp"
,path = "C:/yourpath..."
,species.col= "column with species"
,ID.col="column with patch Id"
,area.col="Column with area"
,dispersal=800#Mean dispersal ability of the species
#(used to generate patch clusters, or components)
)


## End(Not run)
```

`iterate.graph` *Simulate landscape series occupation*

Description

Repeats the process of simulation by `simulate_graph` as many times as required (argument 'iter').

Usage

```
iterate.graph(iter, mapsize, dist_m, areaM, areaSD, Npatch, disp,
span, par1 = "none", par2 = NULL, par3 = NULL, par4 = NULL,
par5 = NULL, method = "percentage", parm, nsew = "none",
succ="none", param_df, kern, conn, colnz, ext, beta1,
b = 1, c1 = NULL, c2 = NULL, z = NULL, R = NULL, graph)
```

Arguments

`iter`	Number of repetitions of the simulation.
`mapsize`	Landscape mosaic side length, in meters. To be internally passed to `rland.graph`.
`dist_m`	Minimum distance between patches (centroid). To be internally passed to `rland.graph`.
`areaM`	Mean area (in hectares). To be internally passed to `rland.graph`.
`areaSD`	SD of the area of patches, in order to give variability to the patches area. To be internally passed to `rland.graph`.
`Npatch`	Number of patches (might be impaired by the dist_m, see the "Note" section). To be internally passed to `rland.graph`.
`disp`	Species mean dispersal ability, in meters. To be internally passed to `rland.graph`.
`span`	Number of time steps (e.g. years) to simulate. To be internally passed to `span.graph`.

par1 — One of the following (default 'none'):

- 'hab' percentage of the number of patches to eliminate.
- 'dincr' minimal distance (between centroids of patches) increase over the simulation (in meters).
- 'darea' percentage of increase/decrease of the mean area of patches, without changing SD.
- 'stoc' simultaneous creation and destruction of patches.
- 'ncsd' simultaneous creation and destruction of patches to the north and south of the landscape.
- 'aggr' correlated habitat destruction.
- 'none' no change.

To be internally passed to span.graph.

par2 — Parameter specifying details for the options in par1: percentage of patches do delete (if par1 = 'hab'); distance, in meters (if par1 = 'dincr'); percentage of increase/decrease of the mean area of patches (if par1 = 'area'); percentage of new patches (if par1 = 'stoc'); 'northerndness' of created patches (if par1 = 'ncsd'); percentage of destroyed patches (if par1 = 'aggr'). To be internally passed to span.graph. Default NULL.

par3 — Additional parameter specifying details for the options in par1: percentage of destroyed patches (if par1 = 'stoc'); 'southerndness' of destroyed patches (if par1 = 'ncsd'); aggregation of destruction (if par1 = 'aggr'). Minimum area for patch deletion, in hectares (if par1='darea'). To be internally passed to span.graph. Default NULL.

par4 — Percentage of created patches (if par1 = 'ncsd'). To be internally passed to span.graph. Default NULL.

par5 — Percentage of destroyed patches (if par1 = 'ncsd'). To be internally passed to span.graph. Default NULL.

method — One of the following (default 'percentage'): **click** - individually select the patches with occurrence of the species by clicking on the map. Use only for individual landscape simulations. However, this option should not be used with iterate.graph. **percentage** - percentage of the patches to by occupied by the species. **number** - number of patches to be occupied by the species. To be internally passed to species.graph.

parm — parameter to specify the species occurrence - either percentage of occupied patches or number of occupied patches, depending on the method chosen. To be internally passed to species.graph.

nsew — 'N', 'S', 'E', 'W' or none - point of entry of the species in the landscape. By default set to "none". To be internally passed to species.graph.

succ — Set the preference of the species for patch successional stage: 'none', 'early', 'mid' and 'late'.

param_df — Parameter data frame delivered by parameter.estimate, including:

- alpha - Parameter relating extinction with distance.
- y - Parameter y in the colonization probability.
- e - Parameter defining the extinction probability in a patch of unit area.

- x - Parameter scaling extinction risk with patch area.

To be internally passed to simulate_graph.

kern	'op1' or 'op2'. Dispersal kernel. See details in the spom function. To be internally passed to spom.
conn	'op1' or 'op2'. Connectivity function. See details in the spom function. To be internally passed to spom.
colnz	'op1', 'op2' or 'op3'. Colonization function. See details in the spom function. To be internally passed to spom.
ext	'op1', 'op2' or 'op3'. Extinction function. See details in the spom function. To be internally passed to spom.
beta1	Parameter affecting long distance dispersal probability (if the Kern='op2'). To be internally passed to spom.
b	Parameter scaling emigration with patch area (if conn='op1' or 'op2'). To be internally passed to spom. By default set to 1.
c1	Parameter scaling immigration with the focal patch area (if conn='op2'). To be internally passed to spom.
c2	Parameter c in the option 3 of the colonization probability (if colnz='op3'). To be internally passed to spom.
z	Parameter giving the strength of the Allee effect (if colnz='op3'). To be internally passed to spom.
R	Parameter giving the strength of the Rescue effect (if ext='op3'). To be internally passed to spom.
graph	TRUE/FALSE, to show graphic output.

Value

Returns a list of five data frames with information regarding the values of mean area, mean inter-patch distance, number of patches occupancy and patch occupancy turnover in each of the iterations, as well as the mean values and SD.

Author(s)

Frederico Mestre and Fernando Canovas

References

References in the spom function.

See Also

rland.graph, span.graph, species.graph, simulate_graph, spom

Examples

```
## Not run:
data(param1)

#Example with 2 iterations (ideally >100):

it1 <- iterate.graph(iter = 2, mapsize =10000, dist_m = 10, areaM = 0.05,
areaSD = 0.02, Npatch = 250, disp = 800, span = 100,
par1 = "hab", par2 = 2, par3 = NULL, par4 = NULL,
par5 = NULL, method = "percentage", parm = 50,
nsew = "none", succ="none", param_df = param1,kern = "op1",
conn = "op1", colnz = "op1", ext = "op1",
beta1 = NULL, b = 1, c1 = NULL, c2 = NULL, z = NULL,
R = NULL, graph =TRUE)

## End(Not run)
```

---

landscape *Class 'landscape'*

Description

Class representing a landscape graph, as produced by rland.graph, convert.graph and import.shape.

Slots

- mapsize - Side of the landscape in meters.
- minimum.distance - Minimum distance between patches centroids, in meters.
- mean.area - Mean patch area in hectares.
- SD.area - Standard deviation of patches area.
- number.patches - Total number of patches.
- dispersal - Species mean dispersal ability, in meters.
- nodes.characteristics - Data frame with patch (node) information (coordinates, area, radius, cluster, distance to nearest neighbor and ID).

Author(s)

Frederico Mestre and Fernando Canovas

landscape_change *Landscape loosing 5% of patches per time step*

Description

This dataset is a list of 100 landscapes with a loss of 5% of each patch's area at each time step. The first landscape is the sample empty landscape.

Format

List of 100 data frames, that represent the evolution of the landscape during 100 time steps.

Examples

```
data(landscape_change)
```

list.stats *Returning information on a dynamic landscape list*

Description

This function allows the computation of some statistics of the sequence of landscapes obtained from simulate.graph. Namely: mean area of the patches, standard deviation of the area, mean pairwise Euclidean distance, total number of patches, species occupation and turnover and mean distance to nearest habitat patch. It allows the graphical representation of the evolution of these statistics.

Usage

```
list.stats(sim_list, stat, plotG)
```

Arguments

sim_list	list from function simulate_graph.
stat	'mean_area', 'sd_area', 'mean_distance', 'n_patches', 'occupation', 'turnover' and 'mean_nneigh'.
plotG	TRUE/FALSE, plot output.

Value

Returns a vector with the evolution of the specified statistics throughout the list of landscapes representing the changes in a dynamic landscape and its occupation. A graphical output is also possible.It is possible to visualize the evolution of mean patch area, standard deviation of the patch area, mean distance between all pairs of patches, number of patches, species percentage of occupation, patch turnover (change in occupational state) and mean distance to nearest habitat patch.

Author(s)

Frederico Mestre and Fernando Canovas

See Also

span.graph

Examples

```
data(rland)
data(landscape_change)
data(param1)

#First, using simulate graph, simulate the occupation on a dynamic landscape
#(output of span.graph):

sim1 <- simulate_graph( rl=rland, rlist=landscape_change, simulate.start=TRUE,
method="percentage", parm=50, nsew="none", succ = "none",
param_df=param1, kern="op1",  conn="op1", colnz="op1",
ext="op1", beta1=NULL, b=1, c1=NULL, c2=NULL, z=NULL, R=NULL)

#Then evaluate species occupancy through the changes suffered by the landscape:

occ <- list.stats(sim_list=sim1, stat="occupation", plotG=TRUE)

#Checking the percentage of occupation in the 40 first landscapes:

head(occ,40)

#Output:

#[1] 50.000000 65.000000 90.000000 96.666667 93.333333 91.666667
#[7] 91.666667 90.000000 93.333333 90.000000 85.000000 83.333333
#[13] 85.000000 88.333333 83.333333 86.666667 81.666667 68.333333
#[19] 70.000000 75.000000 80.000000 73.333333 63.333333 56.666667
#[25] 55.000000 51.666667 46.666667 41.666667 38.333333 21.666667
#[31] 13.333333 13.333333 10.000000  6.666667  5.000000  3.389831
#[37]  1.694915  1.694915  0.000000  0.000000
```

---

manage_expansion_sim *Simulate range expansion simulation*

Description

This function produces dispersal scenarios, considering different habitat networks properties, evaluating the variation in dispersal speed and dispersal maximum distance (of range expansion).

Usage

```
manage_expansion_sim(mapsize, dist_m, areaM, areaSD, Npatch,percI,
                     param, b=1, tsteps, iter, variable,var_min,var_max,by)
```

Arguments

mapsize	Landscape mosaic side length, in meters. To be internally passed to rland.graph
dist_m	Minimum distance between patches (centroid).To be internally passed to rland.graph
areaM	Mean area (in hectares). To be internally passed to rland.graph
areaSD	SD of the area of patches, in order to give variability to the patches area. To be internally passed to rland.graph
Npatch	Number of patches. To be internally passed to rland.graph
percI	Percentage of patch occupancy in the starting landscape. To be internally passed to range_expansion
param	Parameter data frame delivered by parameter.estimate. To be internally passed to range_expansion It includes: • alpha - Parameter relating extinction with distance. • y - Parameter y in the colonization probability. • e - Parameter defining the extinction probability in a patch of unit area. • x - Parameter scaling extinction risk with patch area.
b	Parameter scaling emigration with patch area (if conn='op1' or 'op2') in spom. By default, equal to 1.To be internally passed to range_expansion
tsteps	Number of time steps to simulate (e.g. years).
iter	Number of iterations of the simulation procedure.
variable	Landscape graph characteristic to be varied. One of the following: • area - Mean patch area (in hectares). • dist - Minimum distance between patches (centroid). • npatch - Number of patches. • sizevar - SD of the area of patches.
var_min	Minimum value the changing variable can assume (beware of units used in each case).
var_max	Maximum value the changing variable can assume (beware of units used in each case).
by	Rate of variation of the changing variable.

Details

For details regarding the arguments that are to be internally passed to other functions, see the respective functions. Any of the arguments dist_m, areaM, areaSD, Npatch would be unnecessary if the respective variable is the one to be evaluated (it depends on the parameter variable).

Value

Returns a list of eight data frames. For the first four data frames (NORTH, SOUTH, EAST and WEST) each data frame's first column is the name of the variable to be changed. The other two columns are:

MEAN EXPANSION SPEED
: Expansion speed in each simulated scenario. Speed given in km/time step

MAXIMUM EXPANSION DISTANCE
: Maximum distance of the expanded range, from an occupied site. Given in km.

The other four data frames have detailed information on the simulations for each of the values of parameter "variable". The first column has the distance (in km), and each of the following columns has the time step at which each distance was colonized for each of the simulations.

Warning

This function might be time consuming, and the code is experimental and should be improved in future versions of MetaLandSim.

Author(s)

Frederico Mestre and Fernando Canovas

See Also

rland.graph, range_expansion, expansion

Examples

```
## Not run:

data(param1)


sim_range <- manage_expansion_sim(mapsize=1000, dist_m=0, areaM, areaSD=0.001,
                                  patch=300,percI=50, param=param1, b=1,
                                  tsteps=100, iter=100,variable="area",var_min=0.01,
                                  var_max=0.6,by=0.1)

## End(Not run)
```

`manage_landscape_sim` *Batch landscape simulation*

Description

Runs a series of simulations, using `iterate.graph`, allows changing the simulations parameters in several sequential simulations.

Usage

```
manage_landscape_sim(par_df, parameters_spom, full.output)
```

Arguments

`par_df` Arguments data frame to be used by iterate.graph (each row of this data frame is a set of Arguments). The data frame has to have the following columns in this order (the name of the column is not relevant):

- MDST - Minimum inter-patch distance (in meters).
- NPATCH - Number of patches in the landscape.
- AREA_M - Mean area of the patches (in hectares).
- AREA_SD - SD of the patches' area.
- MAPSIZE - Landscape mosaic side length (in meters).
- SPAN - Number of time steps in the simulation.
- ITER - Number of iterations of the simulation.
- PAR1_SPAN - parm1 for the span.graph function.
- PAR2_SPAN - parm2 for the span.graph function.
- PAR3_SPAN - parm3 for the span.graph function.
- PAR4_SPAN - parm4 for the span.graph function.
- PAR5_SPAN - parm5 for the span.graph function.
- NSEW_SPECIES - Argument nsew for the species.graph function.
- PARM_SPECIES - Argument parm for the species.graph function.
- METHOD_SPECIES - Argument method for the species.graph function.
- KERN - Argument kern for the spom function.
- CONN - Argument conn for the spom function.
- COLNZ - Argument colnz for the spom function.
- EXT - Argument ext for the spom function.
- BETA1 - Argument beta1 for the spom function.
- B - Argument b for the spom function.
- C1 - Argument c1 for the spom function.
- C2 - Argument c2 for the spom function.
- Z - Argument z for the spom function.
- R2 - Argument R for the spom function.
- DISPERSAL - Species mean dispersal ability (in meters).

- SUCCESSION - Species successional preference (early, mid or late).

`parameters_spom`
: Parameters data frame, as given by `parameter.estimate`.

`full.output`
: Creates a folder named 'output' to which it saves the full results of the simulations made with the parameters in each row of 'par_df'. It will generate as many objects as the number of rows in this data frame.

Details

For details regarding the arguments see the respective functions.

Value

Returns a data frame with the parameters used for the simulations and the results (mean occupation, mean number of patches, mean turnover, mean distance and mean area).

Note

Depending on computing capacity, this function can take from several hours to several days to run.

Author(s)

Frederico Mestre and Fernando Canovas

See Also

`rland.graph`, `span.graph`, `species.graph`, `spom`

Examples

```
#Setup the parameters for each simulation:
PAR1_SPAN2 <- rep("ncsd",820)#parameter 1 for the span function
PAR2_SPAN2 <- rep(seq(from=0,to=80,by=2), each=20)#parameter 2 for the span function
PAR3_SPAN2 <- rep(seq(from=0,to=80,by=2),20)#parameter 3 for the span function
PAR4_SPAN2 <- rep(2,820)#parameter 4 for the span function
PAR5_SPAN2 <- rep(2,820)#parameter 5 for the span function
NSEW_SPECIES2 <- rep("none",820)#where to start populating the landscape
PARM_SPECIES2 <- rep(5,820)#parameter for the species function
METHOD_SPECIES2 <- rep("percentage",820)#method for populating the landscape
MAPSIZE2 <- rep(10000,820)#dimension of the landscape
SPAN2 <- rep(100,820)#number of time steps of each simulation
ITER2 <- rep(5,820)#number of iterations of each simulation
NPATCH2 <- rep(800,820)#number of patches
AREA_M2 <- rep(0.45,820)#mean area
AREA_SD2 <- rep(0.2,820)#area sd
MDST2 <- rep(0,820)#minimum distance between
KERN <- rep("op1",820)#kernel
CONN <- rep("op1",820)#connectivity function
COLNZ <- rep("op1",820)#colonization function
EXT <- rep("op1",820)#extinction function
```

```
BETA1 <- rep("NULL",820)
B <- rep(1,820)
C1 <- rep("NULL",820)
C2 <- rep("NULL",820)
Z <- rep("NULL",820)
R2 <- rep("NULL",820)
DISPERSAL2 <- rep(800,820)#mean dispersal ability of the species
SUCC <- rep("early",820)


#Build parameter data frame (keep the order of the parameters):
simulation <- data.frame(MDST2,NPATCH2,AREA_M2,AREA_SD2,
MAPSIZE2,SPAN2,ITER2,PAR1_SPAN2,PAR2_SPAN2,PAR3_SPAN2,PAR4_SPAN2,PAR5_SPAN2,
NSEW_SPECIES2,PARM_SPECIES2,METHOD_SPECIES2,KERN,CONN,COLNZ,EXT,BETA1,B,C1,C2,Z,R2,DISPERSAL2,SUCC)


#Delete vectors used for data frame creation:
rm('PAR1_SPAN2','PAR2_SPAN2','PAR3_SPAN2','PAR4_SPAN2','PAR5_SPAN2',
'NSEW_SPECIES2','PARM_SPECIES2','METHOD_SPECIES2','MAPSIZE2','SPAN2','ITER2',
'NPATCH2','AREA_M2','AREA_SD2','MDST2','KERN','CONN','COLNZ','EXT',
'BETA1','B','C1','C2','Z','R2','DISPERSAL2','SUCC')


## Not run:
data(param1)

ms2 <- manage_landscape_sim(par_df=simulation,parameters_spom=param1)

## End(Not run)
```

`matrix.graph` *Returning a matrix with information on connections between patches*

Description

Based on a landscape graph, this function allows the creation of a matrix of Euclidean distances (straight-line pairwise distance between the margins of all the patches), matrix of topological distances (minimum number of connections between any two patches) and adjacency matrix (this a matrix of 0 and 1, showing the adjacency between any two patches, where 0 means that the patches are not connected and 1 means that the patches are connected).

Usage

```
matrix.graph(rl, mat)
```

Arguments

`rl`	Object of class 'landscape'.
`mat`	mat - one of the following:

- 'euc_distance' - euclidian distance between patches (edge-to-edge).
- 'centr_distance' - euclidian distance between patches (centroid-to-centroid).
- 'adjacency' - adjacency matrix, with values d_ij, taking value 0 if patches i and j are not connected and value 1 if those patches are connected.
- 'top_matrix' - topological distance, with values d_ij, where the value d is the minimum number of connections between the patches i and j. Topological distance is defined as the minimum number of links between patches i and j.

Value

This function returns a matrix (each one of the specified matrices: Euclidean distance, topological distance and adjacency matrix).

Author(s)

Frederico Mestre and Fernando Canovas

See Also

rland.graph

Examples

```
data(rland)

#Computing matrix of topological distances:

matrix.graph(rl=rland, mat="top_matrix")
```

---

mc_df *Modified patch occupancy data of Cabrera vole as a data frame*

Description

One season patch occupancy dataset for *Microtus cabrerae* in SW Portugal (modified). This dataset is in a format directly used by convert.graph and converted to an object class 'metapopulation'.

Usage

```
data(mc_df)
```

Format

A data frame with 685 observations on the following 5 variables.

`ID` Patch Id.

`x` X coordinate.

`y` Y coordinate.

`area` Patch area, in hectares.

`mc` Occupancy state (0/1).

Details

To create this sample dataset the occupancy status of patches was scrambled, however the proportion of occupied patches was kept.

Source

Original field data was obtained during project PERSIST (PTDC/BIA-BEC/105110/2008).

Examples

```
##To be converted in a object of class "metapopulation":
#mc1 <- convert.graph(dframe=mc_df,mapsize=8200,dispersal=800)

data(mc_df)

#Check the columns:

head(mc_df)

#  ID        x       y  area mc
#1  1 1248.254   0.000 0.079  0
#2  2 1420.857  46.725 0.781  1
#3  3 1278.912  52.629 1.053  1
#4  4 6370.625  62.637 0.788  0
#5  5 1151.337  97.140 0.079  0
#6  6 1295.796 104.839 0.137  1
```

MetaLandSim-internal *Internal functions for the MetaLandSim package.*

Description

Internal functions for the MetaLandSim package

Details

These are not to be called by the user.

Source

Coded by Tal Galili. URL: http://www.r-statistics.com/2012/01/merging-two-data-frame-objects-while-preserving-the-rows-order/

`MetaLandSim.GUI` *Graphic User Interface*

Description

User-friendly graphic user interface that allows running the main functions of the package.

Usage

```
MetaLandSim.GUI()
```

Value

Displays the graphic user interface.

Author(s)

Frederico Mestre and Fernando Canovas

See Also

`rland.graph`, `species.graph`, `simulate_graph`, `iterate.graph`, `range_expansion`, `range_raster`

Examples

```
## Not run:
#In order to display the GUI:

MetaLandSim.GUI()

## End(Not run)
```

metapopulation *Class 'metapopulation'*

Description

Class representing a landscape graph with species' patch occupancy data, as produced by `species.graph`, `convert.graph` and `import.shape`.

Slots

- mapsize - Landscape mosaic side length, in meters.
- minimum.distance - Minimum distance between patches centroids, in meters.
- mean.area - Mean patch area in hectares.
- SD.area - Standard deviation of patches area.
- number.patches - Total number of patches.
- dispersal - Species mean dispersal ability, in meters.
- distance.to.neighbours - Data frame with pairwise distance between patches, in meters.
- nodes.characteristics - Data frame with patch (node) information (coordinates, area, radius, cluster, distance to nearest neighbor, ID and species).

Author(s)

Frederico Mestre and Fernando Canovas

metrics.graph *Computes landscape connectivity metrics*

Description

Computes several landscape metrics, mostly derived from graph theory or assuming a graph representation of the landscape.

Usage

```
metrics.graph(rl, metric, dispersal.dist = NULL)
```

Arguments

`rl`	Object of class 'landscape'.
`metric`	one of the following connectivity metrics: • 'NC' - Number of components. • 'LNK' - Number of links connecting the patches. • 'SLC' - Area (in hectares) of the largest group of patches. • 'MSC'- Mean area (in hectares) of the components. • 'HI' - Harary Index. • 'NH' - Normalized Harary Index. • 'ORD' - Landscape (graph) order. • 'GD' - Landscape (graph) diameter. • 'CCP' - Class coincidence probability. • 'LCP' - Landscape coincidence probability. • 'ECS' - Expected cluster size. • 'AWF' - Area-weighted flux. • 'IIC' - Integral index of connectivity. • 'PC' - Probability of connectivity. • 'ECA' - Equivalent connected area.
`dispersal.dist`	Maximum dispersal distance for the binary indexes (NC, LNK, SLC, MSC, HI, NH, ORD, GD, CCP, LCP, ECS, IIC) and mean dispersal distance for the probabilistic indexes (AWF, PC, ECA). When no value is provided the function will assume the dispersal value provided by the 'landscape' object.

Details

These metrics assume different types of links between nodes (patches). Some assume probabilistic connections between nodes (e.g. PC) while others assume binary connections (e.g. NC, SLC, LNK, IIC). Also, these metrics have several degrees of complexity, from the simpler ones (such as NC and LNK) to the more complex (such as IIC and PC). Some are purely structural; the same landscape has the same index whatever the species, while others are measures of functional, where the connectivity of a given landscape is dependent on the species (dispersal ability). Precaution must be taken when looking at the outputs produced by some of these metrics (particularly the simpler, structural ones). Regardless of being simpler to compute, the outputs might be misleading. This metrics can however be used as exploratory tools.
This function was improved by the collaboration of Dr. Santiago Saura (Universidad Politecnica de Madrid).

Detail about each of the metrics:

- 'NC' - Number of components,groups of connected patches, in the landscape graph (Urban and Keitt, 2001). Patches in the same component are accessible, while patches in different components are not connected. More connected landscapes have less components. Threshold dependent (dispersal distance).
- 'LNK' - Number of links connecting the patches (considering that the maximum distance is the species dispersal distance and that these graphs are are binary, which means that nodes are either connected or unconnected) (Pascual-Hortal and Saura, 2006). Higher LNK implies higher connectivity. Threshold dependent (dispersal distance).

- 'SLC' - Area (in hectares) of the largest group of patches, or component (Pascual-Hortal and Saura, 2006). Threshold dependent (dispersal distance).
- 'MSC'- Mean area (in hectares) of a group of patches, or component (Pascual-Hortal and Saura, 2006). Threshold dependent (dispersal distance).
- 'HI' - Harary Index. Originally developed to characterize molecular graphs by Plavsic et al. (1993) it was later transposed to the landscape context by Ricotta et al. (2000). This index was considered by Ricotta et al. (2000) to be more effective from a statistical and ecological perspective. Higher HI implies higher connectivity. Threshold dependent (dispersal distance).
- 'NH' - Normalization of the Harary Index, facilitates analysis because this normalization will set the values between 0 and 1 and allow direct comparison of different habitat networks(Ricotta et al. 2000). Threshold dependent (dispersal distance).
- 'ORD' - Order. Index originated in the graph theory and later translated into the landscape context by Urban and Keitt (2001) provides a simple structural evaluation of the graph: it is the number of patches of the component (group of patches) with more patches. Threshold dependent (dispersal distance).
- 'GD' - Graph diameter. Another index directly derived from graph theory, providing a simple quantification of the graph structure. The graph diameter is the maximum of all the shortest paths between the patches of an habitat network. It is computed in meters (euclidean distance), instead of number of links (such as HI, NH and IIC)(Bunn et al. 2000, Urban and Keith, 2001). Shorter diameter implies faster movement in the habitat network (Minor and Urban, 2008). Threshold dependent (dispersal distance).
- 'CCP' - Class coincidence probability. It is defined as the probability that two randomly chosen points within the habitat belong to the same component. Ranges between 0 and 1 (Pascual-Hortal and Saura 2006). Higher CCP implies higher connectivity. Threshold dependent (dispersal distance).
- 'LCP' - Landscape coincidence probability. It is defined as the probability that two randomly chosen points in the landscape (whether in an habitat patch or not) belong to the same habitat component. Ranges between 0 and 1 (Pascual-Hortal and Saura 2006). Threshold dependent (dispersal distance).
- 'CPL' - Characteristic path length. Mean of all the shortest paths between the network nodes (patches) (Minor and Urban, 2008). The shorter the CPL value the more connected the patches are. Threshold dependent (dispersal distance).
- 'ECS' - Expected cluster size. Mean cluster size of the clusters weighed by area. (O' Brien et al.,2006 and Fall et al, 2007). This represents the size of the component in which a randomly located point in an habitat patch would reside. Although it is informative regarding the area of the component, it does not provide any ecologically meaningful information regarding the total area of habitat, as an example: ECS increases with less isolated small components or patches, although the total habitat decreases(Laita et al. 2011). Threshold dependent (dispersal distance).
- 'AWF' - Area-weighted Flux. Evaluates the flow, weighted by area, between all pairs of patches (Bunn et al. 2000 and Urban and Keitt 2001). The probability of dispersal between two patches (pij), required by the AWF formula, was computed using pij=exp(-k*dij), where k is a constant making pij=0.5 at half the dispersal distance defined by the user. Does not depend on any distance threshold (probabilistic).
- 'IIC' - Integral index of connectivity. Index developed specifically for landscapes by Pascual-Hortal and Saura (2006). It is based on habitat availability and on a binary connection model

(as opposed to a probabilistic). It ranges from 0 to 1 (higher values indicating more connectivity). Threshold dependent (dispersal distance).

- 'PC' - Probability of connectivity. Probability that two points randomly placed in the landscape are in habitat patches that are connected, given the number of habitat patches and the connection probabilities (pij). Similar to IIC, although assuming probabilistic connections between patches (Saura and Pascual-Hortal 2007). Probability of inter-patch dispersal is computed in the same way as for AWF. Does not depend on any distance threshold (probabilistic).
- 'ECA' - The Equivalent Connected Area is the square root of the numerator in PC, not accounting for the total landscape area (AL) (Saura 2011a, 2011b). It is defined as '...the size of a single habitat patch (maximally connected) that would provide the same value of the probability of connectivity than the actual habitat pattern in the landscape' (Saura 2011a).

Value

Returns the numeric value(s), corresponding to the chosen connectivity metric(s) for a given landscape.

Author(s)

Frederico Mestre and Fernando Canovas

References

Bunn, A. G., Urban, D. L., and Keitt, T. H. (2000). Landscape connectivity: a conservation application of graph theory. Journal of Environmental Management, 59(4): 265-278.

Fall, A., Fortin, M. J., Manseau, M., and O' Brien, D. (2007). Spatial graphs: principles and applications for habitat connectivity. Ecosystems, 10(3): 448-461.

Ivanciuc, O., Balaban, T. S., and Balaban, A. T. (1993). Design of topological indices. Part 4. Reciprocal distance matrix, related local vertex invariants and topological indices. Journal of Mathematical Chemistry, 12(1): 309-318.

Laita, A., Kotiaho, J.S., Monkkonen, M. (2011). Graph-theoretic connectivity measures: what do they tell us about connectivity? Landscape Ecology, 26: 951-967.

Minor, E. S., and Urban, D. L. (2007). Graph theory as a proxy for spatially explicit population models in conservation planning. Ecological Applications, 17(6): 1771-1782.

Minor, E. S., and Urban, D. L. (2008). A Graph-Theory Framework for Evaluating Landscape Connectivity and Conservation Planning. Conservation Biology, 22(2): 297-307.

O'Brien, D., Manseau, M., Fall, A., and Fortin, M. J. (2006). Testing the importance of spatial configuration of winter habitat for woodland caribou: an application of graph theory. Biological Conservation, 130(1): 70-83.

Pascual-Hortal, L., and Saura, S. (2006). Comparison and development of new graph-based landscape connectivity indices: towards the priorization of habitat patches and corridors for conservation. Landscape Ecology, 21(7): 959-967.

Plavsic, D., Nikolic, S., Trinajstic, N., and Mihalic, Z. (1993). On the Harary index for the characterization of chemical graphs. Journal of Mathematical Chemistry, 12(1): 235-250.

Ricotta, C., Stanisci, A., Avena, G. C., and Blasi, C. (2000). Quantifying the network connectivity of landscape mosaics: a graph-theoretical approach. Community Ecology, 1(1): 89-94.

Saura, S., and Pascual-Hortal, L. (2007). A new habitat availability index to integrate connectivity in landscape conservation planning: comparison with existing indices and application to a case study. Landscape and Urban Planning, 83(2): 91-103.

Saura, S., Estreguil, C., Mouton, C. & Rodriguez-Freire, M. (2011a). Network analysis to assess landscape connectivity trends: application to European forests (1990-2000). Ecological Indicators 11: 407-416.

Saura, S., Gonzalez-Avila, S. & Elena-Rossello, R. (2011b). Evaluacion de los cambios en la conectividad de los bosques: el indice del area conexa equivalente y su aplicacion a los bosques de Castilla y Leon. Montes, Revista de Ambito Forestal 106: 15-21

Urban, D., and Keitt, T. (2001). Landscape connectivity: a graph-theoretic perspective. Ecology, 82(5): 1205-1218.

See Also

rland.graph

Examples

```
data(rland)

#Compute the Integral index of connectivity of a landscape:

metrics.graph (rl=rland, metric="AWF")
```

`min_distance` *Computes topological distance*

Description

Function to compute topological distance between patches. Topological distance is defined as the minimum number of links between any two patches.

Usage

```
min_distance(rl)
```

Arguments

`rl`	Object of class 'landscape'.

Value

Returns a matrix with the topological distance between the nodes.

Author(s)

Frederico Mestre and Fernando Canovas.

See Also

rland.graph

Examples

```
data(rland)

min_distance(rl=rland)
```

---

occ.landscape *Sample landscape with one simulated occupancy snapshot*

Description

Sample random landscape graph, with species occupancy data (occupancy rate - 50%). Simulated data.

Usage

```
data(occ.landscape)
```

Format

A list with the following elements:

- mapsize - landscape mosaic side length, in meters.
- minimum.distance - minimum distance between patches centroids.
- mean.area - mean area, in hectares.
- SD.area - standard deviation of the area.
- number.patches - number of patches.
- dispersal - mean dispersal ability of the species.
- distance.to.neighbours - data frame with pairwise distance between patches.
- nodes.characteristics - data frame with the characteristics of each patch.

Examples

```
data(occ.landscape)
```

occ.landscape2 *Sample landscape with 10 simulated occupancy snapshots*

Description

Sample species occupancy in a network during 10 time steps. Simulated data.

Usage

```
data(occ.landscape2)
```

Format

A list with the following elements:

- mapsize - landscape mosaic side length, in meters.
- minimum.distance - minimum distance between patches centroids.
- mean.area - mean area, in hectares.
- SD.area - standard deviation of the area.
- number.patches - number of patches.
- dispersal - mean dispersal ability of the species.
- distance.to.neighbours - data frame with pairwise distance between patches.
- nodes.characteristics - data frame with the characteristics of each patch, (species 1 to 10 - occupancy snapshots).

Examples

```
data(occ.landscape2)
```

param1 *Sample parameter data frame number 1*

Description

Sample data frame, as produced by `parameter.estimate`. These parameters are to be passed to `spom`. These are made up parameters, not related to any species.

Usage

```
data(param1)
```

Format

A data frame with 4 rows displaying the four parameters (alpha, x, y, e) to be passed to spom:

- alpha - Parameter relating extinction with distance.
- y - Parameter y in the colonization probability.
- e - Parameter defining the extinction probability in a patch of unit area.
- x - Parameter scaling extinction risk with patch area.

Details

The four parameters are to be passed to spom.

Examples

```
data(param1)

param1

#      par_output
#alpha 0.00100000
#x     0.50000000
#y     2.00000000
#e     0.04662827
```

---

param2 *Sample parameter data frame number 2*

Description

Sample data frame, as produced by parameter.estimate. These parameters are to be passed to spom. These are made up parameters, not related to any species.

Usage

```
data(param1)
```

Format

A data frame with 4 rows displaying the four parameters (alpha, x, y, e) to be passed to spom:

- alpha - Parameter relating extinction with distance.
- y - Parameter y in the colonization probability.
- e - Parameter defining the extinction probability in a patch of unit area.
- x - Parameter scaling extinction risk with patch area.

Details

The four parameters are to be passed to spom.

Examples

```
data(param1)

param1

#      par_output
#alpha 0.00250000
#x     0.50000000
#y     2.00000000
#e     0.04662827
```

`parameter.estimate` *Estimate parameters*

Description

Estimates the parameters of the Stochastic Patch Occupancy Model with the following approaches: regression of snapshot data (Hanski, 1994); Monte Carlo simulation (Moilanen, 1999) and Bayesian MCMC on the full dataset (ter Braak and Etienne, 2003).

Usage

```
parameter.estimate(sp, method, alpha = NULL, nsnap)
```

Arguments

sp	Object of class 'metapopulation' with real patch occupancy data of the focal species.
method	Method to be used in parameter estimation. Available methods: • Rsnap_1 - Regression of snapshot data, using one snapshot (code based on Oksanen, 2004). • Rsnap_x - Regression of snapshot data, using more than one snapshot (code based on Oksanen, 2004). • MCsim - Monte Carlo simulation. • norescue - Bayesian MCMC, not considering Rescue effect. • rescue - Bayesian MCMC, considering Rescue effect.
alpha	Bolean (TRUE/FALSE). Estimate the alpha parameter.
nsnap	Number of snapshots considered.

Details

Parameter alpha describes the effect of distance to dispersal (inverse of the average dispersal distance). Parameter x describes de dependence of the extinction risk on patch size, and consequently on population dimension. Parameter y scales colonization with connectivity. Parameter e is the intrinsic extinction rate of local populations, which is the extinction rate not considering immigration. In the current version the methods 'MCsim', 'rescue' and 'norescue' only create the files to be used in the applications already available. Future versions should allow the direct estimation of parameters without the need for the applications of Moilanen (1999) and Ter Braak and Etienne (2003).
Future versions should include the estimation of other parameters, using the virtual migration model (Hanski et al. 2000).
Regarding the method 'MCsim' the settings file produced (.set) by default has the method Nlr (non-linear regression) chosen. The user should read the file readme.txt, available with the application, where a three step estimation process is described. The objective is to produce the priors for the Monte Carlo simulation to run.
It is highly recommended that the user reads both papers that provide the applications to compute the methods 'MCsim', 'rescue' and 'norescue'. Several editions to the settings and parameters files of both applications might be needed in order to customize the estimation process. This function only generates the input files with the basic needed structure.
Parameter estimation is not the main purpose of this package. As such, the user can estimate the parameters using other available software tools and then apply the estimated parameters in the simulations. The function `create.parameter.df` can be used to create the data frame of the basic spom parameters. Other required parameters can be directly given as arguments to the `iterate.graph`, `spom` or `range_expansion` functions.
The application of the Moilanen paper considers the kernel 'op1', connectivity 'op1', colonization 'op1' and extinction 'op1'. This SPOM (Stochastic Patch Occupancy Model) is known as Incidence Function Model (Hanski,1994 and 1999). In the original version of the mode b=1.However this might be an useful parameter as it scales emigration with patch area. This parameter can be estimated with field data. Moilanen (1998) obtained the value for this parameter by regressing the patch area with known population size.

Value

With the methods 'Rsnap_1' and 'Rsnap_x' eturns a data frame with 4 rows displaying the four parameters (alpha, x, y, e) to be passed to `spom`:

- alpha - Parameter relating extinction with distance.
- y - Parameter y in the colonization probability.
- e - Parameter defining the extinction probability in a patch of unit area.
- x - Parameter scaling extinction risk with patch area.

Regarding the methods 'MCsim', 'rescue' and 'norescue' it returns the files to be used as input in the applications. The files will be saved in the working directory. After running the applications, a data frame can be created in R using the function `create.parameter.df`. This will return a data frame with the same structure as the first two methods.

Note

A vignette is available with detailed information about the computation of the parameters using each method. The method 'MCsim' creates the files (data and settings files) to be used with the application available with the paper by Moilanen (1999). The methods 'rescue' and 'norescue' create the files (data, parameters and distance files)to be used with the application available with the paper by ter Braak and Etienne (2003).
The application by Moilanen is available in http://www.esapubs.org/archive/ecol/E080/003/.
The application by ter Braak and Etienne is available in http://www.esapubs.org/archive/ecol/E084/005/suppl-1.htm.

Author(s)

Frederico Mestre and Fernando Canovas

References

Hanski, I. (1994). A practical model of metapopulation dynamics. Journal of Animal Ecology, 63: 151-162.

Hanski, I. (1999). *Metapopulation Ecology*. Oxford University Press. 313 pp.

Hanski, I., Alho, J. and Moilanen, A. (2000) Estimating the parameters of survival and migration of individuals in metapopulations. Ecology, 81, 239-251.

Moilanen, A. (1998). Long-term dynamics in a metapopulation of the American Pika. The American Naturalist, 152(4), 530-542.

Moilanen, A. (1999). Patch occupancy models of metapopulation dynamics: efficient parameter estimation using implicit statistical inference. Ecology, 80(3): 1031-1043.

Oksanen, J. (2004). Incidence Function Model in R. url.:. http://cc.oulu.fi/~jarioksa/opetus/openmeta/metafit.pdf.

ter Braak, C. J., & Etienne, R. S. (2003). Improved Bayesian analysis of metapopulation data with an application to a tree frog metapopulation. Ecology, 84(1): 231-241.

See Also

create.parameter.df, iterate.graph, range_expansion and spom

Examples

```
data(occ.landscape)

#Using the Regression of snapshot data:

param1 <- parameter.estimate (sp=occ.landscape, method="Rsnap_1")
```

plotL.graph *Plot one landscape of the list created by span.graph*

Description

Plots a given landscape of a landscape sequence from span.graph.

Usage

```
plotL.graph(rl, rlist, nr, species, links, ...)
```

Arguments

rl	Object of class 'landscape'.
rlist	List returned by span.graph.
nr	index of the landscape to display graphically.
species	TRUE/FALSE, TRUE if 'rl' is of class 'metapopulation' or 'FALSE' if rl is of class 'landscape'.
links	TRUE/FALSE, show links between connected patches.
...	Other arguments.

Value

Graphical display of the landscape.

Author(s)

Frederico Mestre and Fernando Canovas

See Also

plot_graph, span.graph, rland.graph

Examples

```
data(rland)
data(landscape_change)

plotL.graph(rl=rland, rlist=landscape_change, nr=50, species=FALSE, links=FALSE)
```

plot_expansion *Graphical display of the expansion*

Description

Plots the expansion object.

Usage

```
plot_expansion(exp)
```

Arguments

exp	Object of class 'expansion'.

Value

Graphical display of the 'expansion' class.

Author(s)

Frederico Mestre and Fernando Canovas

See Also

expansion

Examples

```
data(rg_exp)

plot_expansion(exp=rg_exp)
```

plot_graph *Graphical display of the landscape*

Description

Plots the landscape graph, with or without the species occupation (respectively lists returned by species.graph or rland.graph) and with or without the links between patches.

Usage

```
plot_graph(rl, species, links)
```

Arguments

rl	Object of class 'landscape' (species=FALSE) or 'metapopulation' (species=TRUE).
species	TRUE/FALSE, TRUE if 'x' is of class 'metapopulation' or 'FALSE' if x is of class 'landscape'.
links	TRUE/FALSE, show links between connected patches.

Value

Graphical display of the landscape.

Author(s)

Frederico Mestre and Fernando Canovas

See Also

rland.graph, species.graph

Examples

```
data(rland)
data(occ.landscape)

#Without the species occupancy:
plot_graph(rl=rland, species=FALSE, links=FALSE)

#With the species occupancy:
plot_graph(rl=occ.landscape, species=TRUE, links=FALSE)
```

---

range_expansion *Produce a range expansion model*

Description

This function returns the expansion probability, from a landscape with a given set of parameters, into the four cardinal directions. This can subsequently be converted in a dispersal model by the function range_raster. The dispersal model can be combined with an ecological niche model.

Usage

```
range_expansion(rl, percI, param, b, tsteps, iter)
```

Arguments

rl	Object of class 'landscape'. Starting landscape for the expansion procedure.
percI	Pecentage of patch occupancy in the starting landscape.
param	Parameter data frame delivered by parameter.estimate, including: • alpha - Parameter relating extinction with distance. • y - Parameter y in the colonization probability. • e - Parameter defining the extinction probability in a patch of unit area. • x - Parameter scaling extinction risk with patch area.
b	Parameter scaling emigration with patch area (if conn='op1' or 'op2') in spom. By default, equal to 1.
tsteps	Number of time steps to simulate (e.g. years).
iter	Number of iterations of the simulation procedure.

Details

The expansion algorithm has been improved, since the paper Mestre et al. (2016) describing the package was published. Now, instead of the transition between adjacent landscape units being dictated by the occupation of a spurious node (representing the margin through which the expansion takes place) a somewhat more realistic approach is followed. If, during the metapopulational dynamics simulation, any patch located between the landscape unit (LU) margin and a parallel line placed at a distance equivalent to half of the mean dispersal ability of the species is occupied, than the algorithm assumes that the species will have the ability to go across to the next LU. In this new empty LU initial occupation is defined as follows: a new line is placed, with a spacing equivalent to half the dispersal ability of the species. In the area defined by the margins of the LU and this line the species will occupy in the same proportion as in the preceding LU.
An example, with the expansion eastward (the process is repeated 4 times, one in each cardinal direction): One LU with 200 patches, occupation of 50%, mapsize (length of the LU side) 1000 species mean dispersal ability of 200. Metapopulational dynamics are simulated until one patch is occupied in a area defined by the north, south, east margins of the LU and a vertical line placed at x=800 (1000-200). Then, if any of this patches is occupied, a new LU (a random realization of the same parameter set) is created and initial occupation is defined at an area defined between the west, north and south margins and a line placed at x=200. This occupation level has the same percentage as the previous landscape.

Value

This function returns a list, of class 'expansion', of four data frames with the proportion of occupations at several distances from the closest occupied landscape mosaic. These four data frames correspond to the proportion of occupation to the north, south, east and west. Each data frame has the following columns:

- DISTANCE - Distance (mapsize x number of landscapes).
- OCCUPATION - How many times did the landscape at this distance got occupied by the species (from a total of 'iter' repetitions).
- PROPORTION - Proportion of occupation for the landscape at this distance (OCCUPATION/iter).
- TIME STEP - The average time steps at which a given distance is occupied.

Note

Depending on computing power and number of iterations (parameter 'iter') this function can take from a few hours to several days to run.

Author(s)

Frederico Mestre and Fernando Canovas

See Also

range_raster

Examples

```
## Not run:
#Produce a model of range expansion:
#Note: this function should be run with >100 iterations (parameter "iter").

data(rland)
data(param2)

rg_exp1 <- range_expansion(rl=rland, percI=50, param=param2, b=1, tsteps=100, iter=100)

## End(Not run)
```

---

range_raster *Probability of occupancy, dispersal model*

Description

This function creates the raster map with the expansion simulation, estimating probability of occupancy, at a given time step, based on species dispersal and landscape configuration. range_raster uses the output from range_expansion and a raster map with the species current occupancy.

Usage

```
range_raster(presences.map, re.out, mask.map=NULL, plot.directions=TRUE)
```

Arguments

presences.map	string of the raster file name with species occurrence.
re.out	object of class list expansion. Output from range_expansion.
mask.map	default NULL. String of the raster file name with the mask. Usually, 1 over the area where the analyses should be done.
plot.directions	default TRUE. Whether It will (TRUE) or will not (FALSE) return a graphics for the expansion model functions and raster maps with expansion probabilities in all four cardinal points.

Details

The function automatically reads the raster input files (`presences.map` and `mask.map`, if present). Usually, 0 for absence and 1 for presence in every square cell over a given resolution. Supported file types are those that can be read via rgdal (see gdal. Note that the projection for the raster layer should be one of those supporting metric units (i.e., linear scale is equal in all directions around any point such as Transverse Mercator; see `http://spatialreference.org/`).

Then, it computes and fits single sigmoidal functions for every direction on the expansion movements (building four sub-models, one to each main cardinal direction), as previously computed by `range_expansion`. Four different raster maps are generated (the sub-models), each estimating the probability of expansion for north, south, east and west directions. The four maps are finally summarized into a single range expansion map, which is returned to the user as an object of class `RasterLayer` and saved in the working directory. These four maps do not directly express the probabilities in the output of `range_expansion`. Rather, the outputs are weighted by directionality, so that e.g. the north model favours the dispersal towards the north while penalizing dispersal in every other direction. As such, the resulting dispersal model is not a direct spatial transcription of the four data frames provided by `range_expansion` but an interpretation weighted by the spatial context given by directionality.

Additionally, a raster file is computed, showing the time steps at which each distance is reached. This output depicts the adjustment of a linear model to the output of the four sub-models together, considering the four equally. This output should not be used as guideline to mask the model if running to several time periods. For example if projections to 2050 and 2080 are to be made than the dispersal model should be run twice, adjusting the time steps to the desired date.

This function internally uses a connection to GRASS GIS software through the package `rgrass7-package`, in order to increase the performance for geographical calculations.

Finally, the user might have to manually adjust the starting values of the function `fit.sigmoid`, (defined internally in this function) if it has difficulty adjusting to the output of `range_expansion`.

Value

Produces the spatial realization of the dispersal model, composed by a stack of two objects of the class `RasterLayer` (see `Raster-class` package for further description), with the probability of occupancy and the time step a given distance is occupied. This version of MetaLandSim uses GRASS, version 7 through the package rgrass7. Additionally these rasters are saved in the working directory (files 'PROB' and 'TSTEP') defined by the user and can be directly imported to any GIS software.

Note

This function depends on rgrass7.

Author(s)

Frederico Mestre and Fernando Canovas

References

The same as range_expansion.

See Also

range_expansion, Raster-class, rgrass7, initGRASS

Examples

```
## Not run:

#Installing the rgrass7 development version from GitHub (v. 0.2-1)
#(for now only in GitHub)
library(devtools)
install_github("rsbivand/rgrass7")

#Loading required packages
library(MetaLandSim)
library(rgrass7)

#Loading the range expansion simulation output and required rasters
data(rg_exp)
presences <- system.file("examples/presences.asc", package="MetaLandSim")
mask <- system.file("examples/landmask.asc", package="MetaLandSim")

if (packageVersion("rgrass7") >= "0.2.1") use_sp()

#Initializing a GRASS session in a temporal directory
#(The user should insert the correct path to the executable GRASS file)

#### Under Linux systems:
initGRASS("/usr/bin/grass", home=tempdir())

#### Under Windows systems:
initGRASS("C:/Program Files/GRASS GIS 7.6", home=tempdir())

range.map <- range_raster(presences.map=presences, re.out=rg_exp, mask.map=mask)

#Ploting the results with the raster package
plot(range.map)

#Ploting the results with the rasterVis package
require(rasterVis)
levelplot(range.map, contour=TRUE)


## End(Not run)
```

remove.species *Remove the species occupancy from the landscape*

Description

This function converts an object of class 'metapopulation' (with the species occupancy) in a object of class 'landscape' (without the species occupancy).

Usage

```
remove.species(sp)
```

Arguments

sp	Object of class 'metapopulation'.

Value

Delivers an object of class 'landscape'.

Author(s)

Frederico Mestre and Fernando Canovas

See Also

rland.graph, species.graph

Examples

```
data(occ.landscape)

rl1 <- remove.species(sp=occ.landscape)
```

removepoints *Remove a given number of patches from the landscape*

Description

Randomly removes a given number of patches from the landscape.

Usage

```
removepoints(rl, nr)
```

Arguments

rl	Object of class 'landscape'.
nr	Number of patches to remove.

Value

Returns an object of class 'landscape'.

Author(s)

Frederico Mestre and Fernando Canovas

See Also

rland.graph, addpoints

Examples

```
data(rland)

#Checking the number of patches in the starting landscape:

rland$number.patches

#60

#Removing 10 patches from the landscape:

rl1 <- removepoints(rl=rland, nr=10)

#Checking the number of patches in the output landscape:

rl1$number.patches

#50
```

---

rg_exp *List with range.expansion output*

Description

Output of range_expansion. Object of class 'expansion'.

Usage

```
data(rg_exp)
```

Format

List of four data frames ('NORTH', 'SOUTH', 'EAST' and 'WEST') with the probability of occupations at several distances from the closest occupied landscape mosaic. These four data frames correspond to the probability of occupation to the north, south, east and west. Each data frame has the following columns:

- DISTANCE - Distance (mapsize x number of landscapes).
- OCCUPATION - How many times did the landscape at this distance got occupied by the species (from a total of 'iter' repetitions).
- PROPORTION - Proportion of occupation for the landscape at this distance (OCCUPATION/iter).
- TIME STEP - The average time step during which a given distance is reached.

Examples

```
data(rg_exp)
```

---

rland *Random landscape*

Description

Sample random landscape graph, object of class 'landscape'. It has 60 patches and the landscape mosaic has 1000 meters of side.

Usage

```
data(rland)
```

Format

A list with the following elements:

- mapsize - landscape mosaic side length, in meters.
- minimum.distance - minimum distance between patches centroids).
- mean.area - mean area, in hectares.
- SD.area - standard deviation of the area.
- number.patches - number of patches.
- dispersal - mean dispersal ability of the species.
- nodes.characteristics - data frame with the characteristics of each patch.

Examples

```
data(rland)
```

`rland.graph` *Creates random landscape graph*

Description

One of the key functions of the package, which allows the creation of random landscapes (represented as graphs) with two categories: habitat patch and non-habitat matrix. The landscapes can be different depending on the parameters chosen.

Usage

```
rland.graph(mapsize, dist_m, areaM, areaSD, Npatch, disp, plotG)
```

Arguments

`mapsize`	Landscape mosaic side length, in meters.
`dist_m`	Minimum distance between patches (centroid).
`areaM`	Mean area (in hectares).
`areaSD`	SD of the area of patches, in order to give variability to the patches area.
`Npatch`	Number of patches (might be impaired by the dist_m, see the "Note" section).
`disp`	Species mean dispersal ability, in meters.
`plotG`	TRUE/FALSE, to show graphic output.

Details

The dispersal distance, as given by the parameter 'disp', is used for the computation of some of the connectivity metrics (function `metrics.graph`) and for the graphic representation of the landscapes (in both cases defining the groups of patches, or components). For the simulation of the metapopulational dynamics, the dispersal distance is given through the 'alpha' parameter (the inverse of the mean dispersal ability) in the parameter data frame created by `create.parameter.df`. This has an important consequence: no thresholding (considering the dispersal ability) is assumed when simulating the metapopulational dynamics.

Value

Returns a list, with the following elements:

- mapsizeSide of the landscape in meters.
- minimum.distanceMinimum distance between patches centroids, in meters.
- mean.areaMean patch area in hectares.
- SD.areaStandard deviation of patches area.
- number.patchesTotal number of patches.
- dispersalSpecies mean dispersal ability, in meters.
- nodes.characteristicsData frame with patch (node) information (coordinates, area, radius, cluster, distance to nearest neighbour and ID).
  An additional field, colour, has only graphical purposes.

Note

If the mean distance between patches centroid and the number of patches are both too high then the number of patches is lower than the defined by the user.

Author(s)

Frederico Mestre and Fernando Canovas

See Also

span.graph, species.graph

Examples

```
#Example to create a random landscape graph with 60 patches with a mean area
#of 0.05 hectares.
#The landscape mosaic is a square with 1000 meters side.
#The species mean dispersal ability is 120 meters (in order to connect the patches).
#A plot with the landscape graph is displayed graphically.

rl1 <- rland.graph(mapsize=1000, dist_m=80, areaM=0.05, areaSD=0.02, Npatch=60,
disp=120, plotG=TRUE)
```

---

sim.area *Vector of the areas for each site; here, 100 sites*

Description

By loading simulatedifm this object is loaded.

Format

`sim.area` Vector of the areas for each site; here, 100 sites.

Details

This dataset was created using the code included in **Examples**, in `simulatedifm`.

`sim.det.20` *Array corresponding to nsites x nyears x nvisits*

Description

By loading simulatedifm this object is loaded.

Format

`sim.det.20` 100 x 10 x 3 array corresponding to nsites x nyears x nvisits. Data simulated with year-specific detection probabilities equal to 0.4,0.6,0.2,0.9,0.3,0.4,0.6,0.2,0.9,0.3.

Details

This dataset was created using the code included in **Examples**, in `simulatedifm`.

`sim.distance` *Distance matrix between sampling sites (nsite x nsite).*

Description

By loading simulatedifm this object is loaded.

Format

`sim.distance` nsite x nsite distance matrix.

Details

This dataset was created using the code included in **Examples**, in `simulatedifm`.

`simulatedifm` *Set of simulated data to use with the IFM parameter estimation functions. The data were generated using the code provided in "details".*

Description

This dataset loads several objects:

'sim.area', 'sim.det.20', 'sim.distance', 'z.sim', 'z.sim.20' and 'z.sim.20.fa'.

Format

`sim.area` Vector of the areas for each site; here, 100 sites.

`sim.det.20` 100 x 10 x 3 array corresponding to nsites x nyears x nvisits. Data simulated with year-specific detection probabilities equal to 0.4,0.6,0.2,0.9,0.3,0.4,0.6,0.2,0.9,0.3.

`sim.distance` nsite x nsite distance matrix.

`z.sim` nyear x nsite occupancy data generated with perfect detection.

`z.sim.20` nyear x nsite occupancy data generated with perfect detection with approximately 20% of data missing at random.

`z.sim.20.fa` nyear x nsite occupancy data containing false absences, which can be used to explore the bias of ifm.missing.MCMC and ifm.naive.MCMC when there is imperfect detection.

Details

These datasets were created using the code included in **Examples**.

Examples

```
## Not run:

###############################################
# Areas for 100 hundred sites were created from a log normal distribution with mean
# and variance equal to the mean and variance of the area of the sites in the
# Sierra Foothills black rail population.
# Universal Transverse Mercator locations (UTMs) for each site were
# simulated from the mean and variance of the UTM Northing and Easting
#
# Dynamics were simulated for 1000 years. The parameters were chosen such that the metapopulation
# persisted with reasonable turnover. The last 10 years of these data were retained.
# For the detection data sets, we simulated a removal design based on three visits.

#-------------------
# IFM SIMULATE
#------------------
set.seed(123)

#-------
#AREAS
#-------
mean.log.area=-0.75
sd.log.area=1.33
nsite=100
log.sim.area=rnorm(nsite,mean.log.area,sd.log.area)
sim.area = exp(log.sim.area)

#---------
#DISTANCE
#--------
sim.site.x=rnorm(nsite,643930,9000)
sim.site.y=rnorm(nsite,4340500,10500)
```

```
sim.site.x.mat.col=matrix(rep(sim.site.x,nsite),ncol=nsite,byrow=TRUE)
sim.site.x.mat.row=matrix(rep(sim.site.x,nsite),ncol=nsite)
sim.site.delta.x=(sim.site.x.mat.col-sim.site.x.mat.row)^2

sim.site.y.mat.col=matrix(rep(sim.site.y,nsite),ncol=nsite,byrow=TRUE)
sim.site.y.mat.row=matrix(rep(sim.site.y,nsite),ncol=nsite)
sim.site.delta.y=(sim.site.y.mat.col-sim.site.y.mat.row)^2

# scale distance so that alpha = 2 is reasonable:
sim.distance=((sim.site.delta.x+sim.site.delta.y)^0.5)/100000
diag(sim.distance)=99

#-------------#
# CREATE SPOM #
#-------------#
spom=function(nsite,nyear.sim,alpha,b,y,e,x) {
psi=matrix(rep(NA,nsite*nyear.sim),ncol=nyear.sim)
psi1=rbinom(nsite,1,0.8)
psi[,1]=psi1
s.i.temp=exp(-alpha*sim.distance)
s.i.temp[s.i.temp==1]=0
e.i=e/sim.area^x
e.i[e.i>1]=1

for (i in 2:nyear.sim) {
s.i=s.i.temp
c.i=s.i^2/(s.i^2+y^2)
e.i.re=e.i*(1-c.i)
mu1=psi[,i-1]*(1-e.i.re)+(1-psi[,i-1])*c.i
psi.temp=rbinom(nsite,1,mu1)
psi[,i]=psi.temp
}
psi
}

#---------------
# SIMULATE IFM
#---------------

nyear.sim=1000
psi.sim=spom(nsite,nyear.sim,alpha=20,b=0.5,y=7.5,e=0.25,x=0.25)

nyear=10

# Data from this dataset conforms to the assumptions of IFM Naive:
z.sim=psi.sim[,(nyear.sim+1-nyear):nyear.sim]
apply(z.sim,2,mean)


#CREATE DETECTION HISTORY
p=rep(c(0.4,0.6,0.2,0.9,0.3),2)
nrep=3
```

```
temp=rep(1,nrep*nsite*nyear)
p.mat=matrix(rep(p,nsite),nrow=nsite,byrow=TRUE)
temp.z.sim=z.sim*p.mat
sim.det=rbinom(temp,1,temp.z.sim)
dim(sim.det)=c(nsite,nyear,nrep)

#ENFORCE REMOVAL DESIGN
for (i in 1:nsite) {
  for(t in 1:nyear) {
    if (sim.det[i,t,1]==1) sim.det[i,t,2]=2
    if (sim.det[i,t,1]==1) sim.det[i,t,3]=2
    if (sim.det[i,t,2]==1) sim.det[i,t,3]=2
  }
}

sim.det[sim.det==2]=NA
sim.det.no.missing.values=sim.det

#RANDOMLY CREATE MISSING DATA; 20
# Data are missing when a site was never visited in a given year
unif.mat=runif(nyear*nsite)
z.sim.20=z.sim
z.sim.20[unif.mat<0.2]=NA

sim.det.20=sim.det
sim.det.20[rep(is.na(z.sim.20),nrep)]=NA

#CREATE DATASET WITH MISSING VALUES AND FALSE ABSENCES
#THIS IS TO CHECK HOW IFM.NAIVE AND IFM.MISSING
# LEAD TO BIASES
z.sim.20.fa = apply(sim.det.20,c(1,2),sum,na.rm=TRUE)
z.sim.20.fa[unif.mat<0.20]=NA

# z.sim: Perfect detection, one visit per year.
# z.sim.20: Perfect detection, but 20
# z.sim.20.fa: 20
#   then collapsed to a single observation per year equal to one if a detection occurred.
# sim.det.20: 20
#   The data are arranged in a 3-d array sites x years x visits

save(z.sim,z.sim.20,z.sim.20.fa,sim.det.
20,sim.distance,sim.area,file=paste("SIMULATE_DATA_MDL_CMP",nsite,"_",nyear,
"_",nrep,".RData",sep=""))

## End(Not run)

data(simulatedifm)
ls()
```

---

simulate_graph *Simulate species occupancy in one dynamic landscape*

Description

Simulates the species' occupation on a landscape sequence, resorting to the spom function.

Usage

```
simulate_graph(rl, rlist, simulate.start, method, parm, nsew="none", succ="none",
param_df, kern, conn, colnz, ext, beta1, b, c1, c2, z, R)
```

Arguments

rl	Object of class 'landscape' or 'metapopulation'.
rlist	List delivered by span.graph.
simulate.start	TRUE (rl is of class 'landscape') or FALSE (rl is of class 'metapopulation')
method	One of the following: **click** - individually select the patches with occurrence of the species by clicking on the map. Use only for individual landscape simulations. However, this option should not be used with iterate.graph. **percentage** - percentage of the patches to by occupied by the species. **number** - number of patches to be occupied by the species. To be internally passed to species.graph.
parm	Parameter to specify the species occurrence - either percentage of occupied patches or number of occupied patches, depending on the method chosen. To be internally passed to species.graph.
nsew	'N', 'S', 'E', 'W' or none - point of entry of the species in the landscape. By default set to "none". To be internally passed to species.graph.
succ	Set the preference of the species for patch successional stage: 'none', 'early', 'mid' and 'late'.
param_df	Parameter data frame delivered by parameter.estimate, including: • alpha - Parameter relating extinction with distance. • y - Parameter y in the colonization probability. • e - Parameter defining the extinction probability in a patch of unit area. • x - Parameter scaling extinction risk with patch area. To be internally passed to simulate_graph.
kern	'op1' or 'op2'. Dispersal kernel. See details in the spom function. To be internally passed to spom.
conn	'op1' or 'op2'. Connectivity function. See details in the spom function. To be internally passed to spom.
colnz	'op1', 'op2' or 'op3'. Colonization function. See details in the spom function. To be internally passed to spom.
ext	'op1', 'op2' or 'op3'. Extinction function. See details in the spom function. To be internally passed to spom.
beta1	Parameter afecting long distance dispersal probability (if the Kern='op2'). To be internally passed to spom.
b	Parameter scaling emigration with patch area (if conn='op1' or 'op2'). To be internally passed to spom.

c1	Parameter scaling immigration with the focal patch area (if conn='op2'). To be internally passed to spom.
c2	Parameter c in the option 3 of the colonization probability (if colnz='op3'). To be internally passed to spom.
z	Parameter giving the strength of the Allee effect (if colnz='op3'). To be internally passed to spom.
R	Parameter giving the strength of the Rescue effect (if ext='op3'). To be internally passed to spom.

Value

Returns a list of occupied landscapes, representing the same occupied landscape at different time steps.

Author(s)

Frederico Mestre and Fernando Canovas

See Also

spom, span.graph, rland.graph, iterate.graph

Examples

```
data(rland)
data(landscape_change)
data(param1)

sim1 <- simulate_graph(rl=rland,
rlist=landscape_change,
simulate.start=TRUE,
method="percentage",
parm=50,
nsew="none",
succ = "none",
param_df=param1,
kern="op1",
conn="op1",
colnz="op1",
ext="op1",
beta1=NULL,
b=1,
c1=NULL,
c2=NULL,
z=NULL,
R=NULL
)
```

`span.graph` *Simulate landscape dynamics over a number of time steps*

Description

This function gets an initial landscape graph and gradually applies changes. For a good review and classification of such changes see Bogaert et al. (2004) (not all described changes have been applied here). Future versions of the package should include other methods to change the landscape.

Usage

```
span.graph(rl, span = 100, par1 = 'none', par2 = NULL,
           par3 = NULL, par4 = NULL, par5 = NULL)
```

Arguments

`rl`	Object of class 'landscape'.
`span`	Number of time steps (e.g. years) to simulate.
`par1`	Parameter determining the dynamism type. One of the following (default 'none'):

- 'hab' percentage of the number of patches to eliminate.
- 'dincr' minimal distance (between centroids of patches) increase over the simulation (in meters).
- 'darea' percentage of increase/decrease of the mean area of patches, without changing SD. Patches with area <1 square meter are deleted.
- 'stoc' simultaneous creation and destruction of patches with variation in the number of created and destroyed patches.
- 'stoc2' simultaneous creation and destruction of patches with the same percentage of created and destroyed patches derived from the number of patches of the landscape in the preceding time step.
- 'ncsd' simultaneous creation and destruction of patches to the north and south of the landscape.
- 'aggr' correlated habitat destruction.
- 'none' no change.

The percentage of patches to be generated or destroyed at each time step is not fixed (except for 'stoc2' in which case the percentage of created and destroyed patches is the same and directly computed from the number of patches in the preceeding time step, allowing to have landscape dynamism without change in the number of patches). For example if the landscape at the time step t-1 has 200 patches and the user wishes to set up a destruction rate of 5%, than the number of destroyed patches is given by a random number obtained from a Poisson distribution with mean 10 (5% of 200).

`par2`	Parameter specifying details for the options in par1: percentage of patches do delete (if par1='hab'); distance, in meters (if par1='dincr'); percentage of increase/decrease (increase with negative sign) of the mean area of patches (if

	par1='darea'); percentage of created/destroyed patches (if par1='stoc'); percentage of created patches (if par1='stoc2'); 'northerndness' of created patches (if par1='ncsd'); percentage of destroyed patches (if par1='aggr').
par3	Additional parameter specifying details for the options in par1: percentage of destroyed patches (if par1='stoc2'); 'southerndness' of destroyed patches (if par1='ncsd'); aggregation of destruction (if par1='aggr'). Minimum area for patch deletion, in hectares (if par1='darea').
par4	Percentage of created patches (if par1='ncsd').
par5	Percentage of destroyed patches (if par1='ncsd').

Value

Returns a list of data frames with the nodes characteristics of a given number of landscapes that suffer a specified change. The fields of these data frames are the same as those from the nodes characteristics resulting from rland.graph.

Author(s)

Frederico Mestre and Fernando Canovas

References

Bogaert, J., Ceulemans, R., & Salvador-Van Eysenrode, D. (2004). Decision tree algorithm for detection of spatial processes in landscape transformation. Environmental Management, 33(1): 62-73.

See Also

rland.graph, simulate_graph, iterate.graph

Examples

```
data(rland)

#Simulating a decrease of 5% in the number of patches through 100 time steps:

span1 <- span.graph(rl=rland, span=100, par1="hab", par2=5, par3=NULL, par4=NULL, par5=NULL)
```

---

species.graph *Simulate landscape occupation*

Description

Given a set of parameters, this function allows to simulate the occupation of an empty landscape, class "metapopulation".

Usage

```
species.graph(rl, method = 'percentage', parm, nsew = 'none', plotG = TRUE)
```

Arguments

rl	Object of class "landscape".
method	One of the following (default 'percentage'): **click** - individually select the patches with occurrence of the species by clicking on the map. Use only for individual landscape simulations. **percentage** - percentage of the patches to be occupied by the species. **number** - number of patches to be occupied by the species.
parm	Parameter to specify the species occurrence - either percentage of occupied patches or number of occupied patches, depending on the method chosen.
nsew	'N', 'S', 'E', 'W' or none - point of entry of the species in the landscape. By default set to "none".
plotG	TRUE/FALSE, to show graphic output.

Value

Returns a list, with the following elements:

- mapsize - Landscape mosaic side length, in meters.
- minimum.distance - Minimum distance between patches centroids, in meters.
- mean.area - Mean patch area in hectares.
- SD.area - Standard deviation of patches area.
- number.patches - Total number of patches.
- dispersal - Species mean dispersal ability, in meters.
- distance.to.neighbours - Data frame with pairwise distance between patches, in meters.
- nodes.characteristics - Data frame with patch (node) information (coordinates, area, radius, cluster, distance to nearest neighbour, ID and species).

An additional field, colour, has only graphical purposes.

Author(s)

Frederico Mestre and Fernando Canovas

See Also

rland.graph, simulate_graph, remove.species

Examples

```
data(rland)

##Creating a 50% occupation in an empty landscape (using the "landscape" dataset):

sp1 <- species.graph(rl=rland, method="percentage", parm=50, nsew="none", plotG=TRUE)
```

spom *Stochastic Patch Occupancy Model*

Description

This function predicts the occupancy status of each patch in a landscape in the time step t+1, based on the occupancy information on time step t.

Usage

```
spom(sp, kern, conn, colnz, ext, param_df,
     beta1 = NULL, b = 1, c1 = NULL, c2 = NULL,
     z = NULL, R = NULL, succ="none", max_age=1)
```

Arguments

sp	Landscape with species occupancy, object of class 'metapopulation'.
kern	'op1' or 'op2'. Dispersal kernel. See details.
conn	'op1' or 'op2'. Connectivity function. See details.
colnz	'op1', 'op2' or 'op3'. Colonization function. See details.
ext	'op1', 'op2' or 'op3'. Extinction function. See details.
param_df	Parameter data frame delivered by parameter.estimate, including: • alpha - Parameter relating extinction with distance. • y - Parameter y in the colonization probability. • e - Parameter defining the extinction probability in a patch of unit area. • x - Parameter scaling extinction risk with patch area.
beta1	Parameter affecting long distance dispersal probability (if the Kern='op2').
b	Parameter scaling emigration with patch area (if conn='op1' or 'op2'). By default set to 1.
c1	Parameter scaling immigration with the focal patch area (if conn='op2').
c2	Parameter c in the option 3 of the colonization probability (if colnz='op3').
z	Parameter giving the strength of the Allee effect (if colnz='op3').
R	Parameter giving the strength of the Rescue effect (if ext='op3').
succ	Set the preference of the species for patch successional stage: 'none', 'early', 'mid' and 'late'.
max_age	Default value set to 1. This argument should not be changed by the user. It is used only when the function runs inside others.

Details

In order to visualize which parameter combination is valid for each option, please refer to the following table (alpha, x, y and e are delivered by parameter.estimate, as a data frame):

parameter	kern_1	kern_2	conn_1	conn_2	colnz_1	colnz_2	colnz_3	ext_1	ext_2	ext_3
alpha	x	x								
x								x	x	x
y					x	x				
e								x	x	x
beta1		x								
b			x	x						
c1				x						
c2							x			
z							x			
R										x

A Stochastic Patch Occupancy Model (SPOM) is a type of model which models the occupancy status of the species on habitat patches as a Markov chain (Moilanen, 2004). These models are a good compromise between capturing sufficient biological detail and being easy to parametrize with occupancy data. With SPOMs it is possible to predict the probability of extinction or colonization of every patch in a landscape, given the current occupancy state of all the patches (Etienne et al. 2004).

Dispersal Kernel

Option 1 (Hanski, 1994 and 1999)

$$D(D_{ij}, \alpha) = exp(-\alpha.d_{ij})$$

Option 2 (Shaw, 1995)

$$D(D_{ij}, \alpha, \beta) = \frac{1}{1 + \alpha.d_{ij}^{\beta}}$$

where *dij is the distance between patches i and j.*

- Option 1 - Negative exponential. Earlier studies (until the end of the 1990) frequently used this type of thin-tailed kernels (Nathan et al. 2012).
- Option 2 - Fat-tailed kernel. The shape of the dispersal kernel is highly significant only when the metapopulation consists of several moderately small patch clusters, which are relatively far from each other. In this kind of a system, a patch cluster may go extinct, and long-distance dispersal will be important in determining the recolonization probability of the empty cluster (Shaw, 1995 and Moilanen, 2004). This type of fat-tailed kernels has become more frequent in recent works (Nathan et al. 2012). For

  $$\beta = 2$$

  this is the Cauchy distribution.

Connectivity

Option 1 (Moilanen, 2004)

$$S_i = \sum pj.D(d_{ij}, \alpha).A_j^b$$

Option 2 (Moilanen and Nieminen, 2002)

$$S_i = A_i^c \sum p_j.D(d_{ij}, \alpha).A_j^b$$

where *Ai and Aj are the areas of patches i(focal patch) and j(other patches), respectively; dij is the distance between patches i and j and pj is the occupation status (0/1) of patch j*

- Option 1 - In the version of Hanski (1994), de kernel is the negative exponential (option 1) and b is set to 1. In this more flexible version, the parameter b scales emigration with patch area (Moilanen, 2004).
- Option 2 - In Moilanen & Nieminen (2002) the kernel is the negative exponential (option 1). This metric considers the value of the focal patch's area, which was found to provide better results by Moilanen & Nieminen (2002), being less sensitive to errors in the estimation of a. Parameters b and c scale, respectively emigration and immigration, as a function of patch area (focal patch in the case of c). See 'note'.

Colonization function

Option 1 (Hanski, 1994, 1999)

$$C_i = \frac{S_i^2}{S_i^2 + y^2}$$

Option 2 (Moilanen, 2004)

$$C_i = 1 - exp(-y.S_i)$$

Option 3 (Ovaskainen, 2002)

$$C_i = \frac{S_i^z}{S_i^z + \frac{1}{c}}$$

where *Si is connectivity.*

- Option 1 - It's the first version of the colonization probability, it includes Allee effect (however the strength of this effect cannot be modified) Hanski (1994). Colonization probability is defined as a sigmoid function of the connectivity of patch i.
- Option 2 - This option assumes that immigrating individuals originate colonization events independently, therefore, with no Allee effect. Adequate for species (plants) with passive dispersal (Moilanen, 2004).
- Option 3 - Here, as in option 1, the colonization probability is defined as a sigmoid function of the connectivity of patch i, and the user can change the strength of the Allee effect, by changing the parameter z, with values >1 reflecting the presence of this effect (Ovaskainen, 2002). In the original version of the IFM (option 1) Hanski (1994) assumed a relatively strong Allee effect (z=2). Parameter c describes the species ability to colonize (Ovaskainen & Hanski, 2001 and Ovaskainen ,2002).

Extinction function

Option 1 (Hanski, 1994, 1999)

$$E_i = min(1, \frac{e}{A_i^x})$$

Option 2 (Hanski and Ovaskainen, 2000 and Ovaskainen and Hanski, 2002)

$$E_i = 1 - exp(\frac{-e}{A_i^x})$$

Option 3 (Ovaskainen, 2002)

$$E_i = min[1, \frac{e}{A_i^x}.(1 - C_i)^R]$$

where *Ai is the area of the focal patch and Ci is the colonization probability of the focal patch.*

- Option 1 - Original version developed by Hanski (1994).
- Option 2 - Used e.g. in the spatially realistic Levins model (Hanski & Ovaskainen, 2000 and Ovaskainen & Hanski, 2002). Parameter x scales extinction probability with patch area.
- Option 3 - Same as option 1, but considering the Rescue effect (with the strength of this effect being given by R). If R=0 there is no Rescue effect, however, if R>0, the Rescue effect grows exponentially with the probability of not being colonized. In the original version of this function Hanski (1994) assumed R=1.

Here, parameter *x* defines de degree to which the extinction rate is sensitive to the patch area. If *x>1*, with the increase of *Ai* the extinction rate rapidly approximates zero. The populations in the larger patches becomes almost impossible to extinguish. However, if *x* is small the extinction rate decreases slower with increasing *Ai*.

Value

Delivers a list similar to the class 'metapopulation' but with two additional columns in the data frame nodes.characteristics: 'species2'(which is the occupation in the next time step) and turn (turnover between occupancies).

Note

Future versions of the package should include the virtual migration model (Hanski et al. 2000), which allows the estimation of migration related parameters (relevant to the option 2 of connectivity).

Author(s)

Frederico Mestre and Fernando Canovas

References

Etienne, R. S., ter Braak, C. J., and Vos, C. C. (2004). Application of stochastic patch occupancy models to real metapopulations. In Hanski, I. and Gaggiotti, O.E. (Eds.) Ecology, Genetics, and Evolution of Metapopulations. Elsevier Academic Press. 696 pp.

Hanski, I. (1994). A practical model of metapopulation dynamics. Journal of Animal Ecology, 63: 151-162.

Hanski, I. (1999). *Metapopulation Ecology*. Oxford University Press. 313 pp.

Hanski, I., Alho, J., and Moilanen, A. (2000). Estimating the parameters of survival and migration of individuals in metapopulations. Ecology, 81(1), 239-251.

Hanski, I., and Ovaskainen, O. (2000). The metapopulation capacity of a fragmented landscape. Nature, 404: 755-758.

Moilanen, A. (2004). SPOMSIM: software for stochastic patch occupancy models of metapopulation dynamics. Ecological Modelling, 179(4), 533-550.

Moilanen, A., and Nieminen, M. (2002). Simple connectivity measures in spatial ecology. Ecology, 83(4): 1131-1145.

Nathan, R., Klein, E., Robledo-Arnuncio, J.J. and Revilla, E. (2012). Dispersal kernels: review. in Clobert, J., Baguette, M., Benton, T. and Bullock, J.M. (Eds.) Dispersal Ecology and Evolution. Oxford University Press. Oxford, UK. 462 pp.

Ovaskainen, O. (2002). The effective size of a metapopulation living in a heterogeneous patch network. The American Naturalist: 160(5), 612-628.

Ovaskainen, O. and Hanski, I. (2001). Spatially structured metapopulation models: global and local assessment of metapopulation capacity. Theoretical Population Biology, 60(4), 281-302.

Ovaskainen, O., and Hanski, I. (2002). Transient dynamics in metapopulation response to perturbation. Theoretical Population Biology, 61(3): 285-295.

Ovaskainen, O. and Hanski, I. (2004). Metapopulation dynamics in highly fragmented landscapes. In Hanski, I. & Gaggiotti, O.E. (Eds.) Ecology, Genetics, and Evolution of Metapopulations. Elsevier Academic Press. 696 pp.

Shaw, M.W., (1995). Simulation of population expansion and spatial pattern when individual dispersal distributions do not decline exponentially with distance. Proc. R. Soc. London B: 259, 243-248.

See Also

species.graph, simulate_graph, iterate.graph

Examples

```
data(occ.landscape)
data(param1)

#Simulating the occupation in the next time step:

landscape2 <- spom(sp=occ.landscape,
kern="op1",
conn="op1",
colnz="op1",
ext="op1",
param_df=param1,
beta1=NULL,
b=1,
c1=NULL,
c2=NULL,
z=NULL,
R=NULL,
```

```
succ="none"
)

#The output has two new columns in the data frame nodes.characteristics: species2
#(occupation in the next time step) and turn (turnover - change of occupation status,
#1 if changed and 0 if not).:

head(landscape2)

#         x         y      areas    radius cluster    colour nneighbour
#1 718.5011 228.47190 0.05741039 13.518245       1 #FF0000FF   91.80452
#2 494.3624  73.29165 0.08755563 16.694257       1 #FF0000FF   98.98432
#3 809.2326 245.90046 0.09384384 17.283351       1 #FF0000FF  166.68205
#4 638.8057 149.35122 0.08858989 16.792569       1 #FF0000FF   82.60306
#5 874.2010  19.78104 0.03621793 10.737097       1 #FF0000FF   92.26625
#6 605.3937  70.34944 0.03066018  9.878987       1 #FF0000FF  131.22261
#  ID species species2 turn
#1  1       1        1    0
#2  2       0        1    1
#3  3       1        1    0
#4  4       0        0    0
#5  5       0        1    1
#6  6       1        1    0
```

---

summary_landscape *Summarize 'landscape' class objects*

Description

This function summarizes a landscape class object.

Usage

```
summary_landscape(object)
```

Arguments

object	Object of class landscape

Details

This function can be used to retrieve basic information on the objects of class 'landscape'.

Value

Returns a data frame with the following information on a landscape class object:

landscape area (hectares)
: Landscape mosaic area, in hectares

`number of patches`
: Number of patches in the landscape

`mean patch area (hectares)`
: Mean patch area, in hectares

`SD patch area`
: SD of the patch area

`mean distance amongst patches (meters)`
: Mean inter-patch distance, in meters

`minimum distance amongst patches (meters)`
: Minimum inter-patch distance, in meters

Note

The minimum distance between patches is different from that given in the object of class 'landscape', in the slot 'minimum.distance'. This is because this output is computed from the landscape structure and the one in the 'landscape' object was the parameter used to built the landscape. The minimum inter-patch distance given as a parameter in the function `rland.graph` will consider distance between patch centroids. The minimum inter-patch distance returned here considers the edge-to-edge distance, so this might be smaller that the parameter of `rland.graph`. In order to see the difference between centroid-to-centroid and edge-to-edge inter-patch distance compute both using the `matrix.graph` function (methods are 'centr_distance' and 'euc_distance', respectively).

Author(s)

Frederico Mestre and Fernando Canovas

See Also

`rland.graph`, `landscape`, `matrix.graph`

Examples

```
data(rland)

summary_landscape(object=rland)

#                                          Value
#landscape area (hectares)                100.000
#number of patches                         60.000
#mean patch area (hectares)                0.061
#SD patch area                             0.041
#mean distance amongst patches (meters)    528.345
#minimum distance amongst patches (meters)  51.780
```

summary_metapopulation *Summarize 'metapopulation' class objects*

Description

This function summarizes a metapopulation class object.

Usage

```
summary_metapopulation(object)
```

Arguments

object	Object of class metapopulation

Details

This function can be used to retrieve basic information on the objects of class 'metapopulation'.

Value

Returns a data frame with the following information on a metapopulation class object:

landscape area (hectares)
: Landscape mosaic area, in hectares

number of patches
: Number of patches in the landscape

mean patch area (hectares)
: Mean patch area, in hectares

SD patch area
: SD of the patch area

mean distance amongst patches (meters)
: Mean inter-patch distance, in meters

minimum distance amongst patches (meters)
: Minimum inter-patch distance, in meters

species occurrence - snapshot
: Occupation data of the focal species, numbered from 1 to the number of snapshots

Note

The minimum distance between patches is different from that given in the object of class 'landscape', in the slot 'minimum.distance'. This is because this output is computed from the landscape structure and the one in the 'landscape' object was the parameter used to built the landscape. The minimum inter-patch distance given as a parameter in the function rland.graph will consider distance between patch centroids. The minimum inter-patch distance returned here considers the edge-to-edge distance, so this might be smaller that the parameter of rland.graph. In order to see the

difference between centroid-to-centroid and edge-to-edge inter-patch distance compute both using the matrix.graph function (methods are 'centr_distance' and 'euc_distance', respectively).

Author(s)

Frederico Mestre and Fernando Canovas

See Also

species.graph, metapopulation, matrix.graph

Examples

```
data(occ.landscape)
data(occ.landscape2)

summary_metapopulation(object=occ.landscape)

#                                          Value
#landscape area (hectares)                100.000
#number of patches                         60.000
#mean patch area (hectares)                 0.061
#SD patch area                              0.041
#mean distance amongst patches (meters)    528.345
#minimum distance amongst patches (meters)  51.780
#species occurrence - snapshot 1            50.000


summary_metapopulation(object=occ.landscape2)

#                                          Value
#landscape area (hectares)                100.000
#number of patches                         60.000
#mean patch area (hectares)                 0.069
#SD patch area                              0.039
#mean distance amongst patches (meters)    521.717
#minimum distance amongst patches (meters)  45.905
#species occurrence - snapshot 1            50.000
#species occurrence - snapshot 2            58.333
#species occurrence - snapshot 3            61.667
#species occurrence - snapshot 4            61.667
#species occurrence - snapshot 5            58.333
#species occurrence - snapshot 6            60.000
#species occurrence - snapshot 7            70.000
#species occurrence - snapshot 8            68.333
#species occurrence - snapshot 9            68.333
#species occurrence - snapshot 10           56.667
```

z.sim — *Occupancy data generated with perfect detection.*

Description

By loading simulatedifm this object is loaded.

Format

`z.sim` nyear x nsite occupancy data generated with perfect detection.

Details

This dataset was created using the code included in **Examples**, in `simulatedifm`.

z.sim.20 — *Occupancy data generated with perfect detection with approximately 20% of data missing at random.*

Description

By loading simulatedifm this object is loaded.

Format

`z.sim.20` nyear x nsite occupancy data generated with perfect detection with approximately 20% of data missing at random.

Details

This dataset was created using the code included in **Examples**, in `simulatedifm`.

z.sim.20.fa — *Occupancy data containing false absences*

Description

By loading simulatedifm this object is loaded.

Format

`z.sim.20.fa` nyear x nsite occupancy data containing false absences, which can be used to explore the bias of ifm.missing.MCMC and ifm.naive.MCMC when there is imperfect detection.

Details

This dataset was created using the code included in **Examples**, in `simulatedifm`.

Index