Package 'briskaR'

October 12, 2022

Type Package

Encoding UTF-8

Title Biological Risk Assessment

Version 1.0.4

Date 2021-12-07

Author Virgile Baudrot [aut],
Emily Walker [aut],
Jean-Francois Rey [aut, cre],
Melen Leclerc [aut],
Samuel Soubeyrand [ctb],
Marc Bourotte [ctb]

Maintainer Jean-Francois Rey <jean-francois.rey@inrae.fr>

Description A spatio-temporal exposure-hazard model for assessing biological risk and impact. The model is based on stochastic geometry for describing the landscape and the exposed individuals, a dispersal kernel for the dissemination of contaminants, a set of tools to handle spatio-temporal dataframe and ecotoxicological equations.
Walker E, Leclerc M, Rey JF, Beaudouin R, Soubeyrand S, and Messean A, (2017), A Spatio-Temporal Exposure-Hazard Model for Assessing Biological Risk and Impact, Risk Analysis, <doi:10.1111/risa.12941>.
Leclerc M, Walker E, Messean A, Soubeyrand S (2018), Spatial exposure-hazard and landscape models for assessing the impact of GM crops on non-target organisms, Science of the Total Environment, 624, 470-479.

URL https://gitlab.paca.inrae.fr/biosp/briskaR

BugReports https://gitlab.paca.inrae.fr/biosp/briskaR/-/issues

License GPL (>= 2) | file LICENSE

LazyData True

BuildVignettes True

NeedsCompilation yes

VignetteBuilder knitr

Depends methods, grDevices (>= 3.0.0), graphics (>= 3.0.0), stats (>= 3.0.2), R (>= 3.0.2)

Imports deldir(>= 0.1), deSolve, fasterize, fftwtools(>= 0.9.6), MASS(>= 7.3.29), mvtnorm(>= 1.0.2), raster(>= 2.3.0), Rcpp (>= 1.0.0), rgdal (>= 0.9), rgeos(>= 0.3), sf (>= 0.7-1), sp (>= 1.0-17)

LinkingTo testthat (>= 3.0.0), Rcpp, RcppArmadillo

Suggests ggplot2, knitr, rmarkdown, dplyr, testthat (>= 3.0.1), xml2

RoxygenNote 7.1.2

Repository CRAN

Date/Publication 2021-12-07 10:30:02 UTC

R topics documented:

`briskaR-package` *Biological Risk Assessment*

Description

A spatio-temporal exposure-hazard model for assessing biological risk and impact. The model is based on stochastic geometry for describing the landscape and the exposed individuals, a dispersal kernel for the dissemination of contaminants and an ecotoxicological equation.

Details

The briskaR package contains functions and methods for quantifying spatio-temporal variation in contamination risk around known polygon sources of contaminants, and quantifies the impact of the contaminants on the surrounding population of individuals which are located in habitat areas and are susceptible to the contaminants.

The package implements an spatio-temporal exposure-hazard model based on (i) tools of stochastic geometry (marked polygon and point processes) for structuring the landscape and describing the location of exposed individuals, (ii) a method based on a dispersal kernel describing the dissemination of contaminant particles from polygon sources, and (iii) ecotoxicological equations describing how contaminants affect individuals of the exposed population.

Author(s)

Virgile Baudrot <virgile.baudrot@posteo.fr>

Emily Walker <emily.walker@inrae.fr>

Jean-Francois Rey <jean-francois.rey@inrae.fr>

Melen Leclerc <melen.leclerc@inrae.fr>

Samuel Soubeyrand <Samuel.Soubeyrand@inrae.fr>

Marc Bourotte <marc.bourotte@inrae.fr>

Maintainer: Jean-Francois REY <jean-francois.rey@inrae.fr>

See Also

Useful links:

- https://gitlab.paca.inrae.fr/biosp/briskaR
- Report bugs at https://gitlab.paca.inrae.fr/biosp/briskaR/-/issues

briskaRLoadInternProjection *Load an internal working projection PROJ.4*

Description

Will load a projection as internal package working projection

Usage

```
briskaRSetInternProjection(proj = LAMBERT_93)
```

Arguments

proj	A character string of projection arguments, must be in the PROJ.4 documentation

brk_addFD *Functional DATA*

Description

Add time series to sf objects

Usage

```
brk_addFD(sf, key, FUN, ...)

brk_addFD2(sf, keyConstraint, key, FUN, ...)
```

Arguments

sf	object of class sf. See sf for details.
key	the name of the new column, as strings or symbols
FUN	the function to be applied to each element of sf. In the case of functions like +, backquoted or quoted. See lapply functions for details.
...	optional arguments to FUN. See lapply functions for details.
keyConstraint	character string. The reference of the column to be checked

Value

A sf object with addition functional data feature (or feature dynamic).

brk_cFilterFD *Function used to filter functional data in sf objects.*

Description

Function used to filter functional data in sf objects.

Usage

```
brk_cFilterFD(sf, key, index)

brk_cFilterFD_(sf, key, index)

brk_cFilterFD2(sf, key1, key2, index)

brk_cFilterFD3(sf, key1, key2, key3, index)
```

Arguments

sf	sf. An object of class sf
key	character string. The name of the column to select
index	integer (or vector). The index of the functional data.
key1	character string. The name of the column to select
key2	character string. The name of the column to select
key3	character string. The name of the column to select

brk_dispersal *Compute dispersal*

Description

Simulate contaminants or individuals frequency over the landscape by two steps: dispersal of elements and local intensity/frequency of elements after dispersal.

Usage

```
brk_dispersal(
  object,
  size_raster,
  tolerance_square,
  kernel,
  kernel.options,
  nbr_cores,
  squared_frame
)
```

Arguments

`object`	sf or patialPolygonsDataFrame. A simple feature of class sf or SpatialPolygons-DataFrame
`size_raster`	integer. Raster size (default = 2^10)
`tolerance_square`	numeric. Tolerance rate to test if an sf set is squared
`kernel`	string. Dispersion kernel, function name (default = NIG)
`kernel.options`	list. Parameters list for the kernel function
`nbr_cores`	integer. Parameters for parallel computing: the number of cores to use, i.e. at most how many child processes will be run simultaneously. Default is 1 (non parallel).
`squared_frame`	sf. Select the sf to be considered as frame to rasterized. Default is 'NULL', and 'object' is used.

Details

The dispersal of contaminants or individuals is implemented by rastering the landscape and by computing the convolution between sources emissions and a dispersal kernel.

The dispersion kernel by default is Normal Inverse Gaussian kernel ("NIG" function). Currently, two others are implemented "geometric" (with parameter `a`) and "2Dt" kernels (with parameters `a`, `b`, `c1`, `c2`).

`brk_emission` *Wrapper function brk_emission*

Description

'brk_emission' This function simulates emissions. Will simulate emissions shape in source fields of a landscape.

'brk_emission_landscape' Simulate sources emission intensity. With: - 'emission' is the quantity of emission per time unit per spatial unit. With other argument, we have: - 'emission = density x intensity = density x production x intensity_pmf'. - 'density' is the density of the source, so the quantity of source per spatial unit. - 'intensity' is the intensity of the emission for 1 source: that is the quantity of emission per time unit per source. - 'production' is the overall production, the total emission, for one source unit: quantity of emission for the period - 'intensity_pmf' is the distribution of emission along time. So we have 'intensity = production x intensity_pmf'.

Usage

```
brk_emission(sf, keyTime, key, FUN)

brk_emission_landscape(
  sf,
  timeline = 1:61,
```

```
  emission = NULL,
  density = NULL,
  intensity = NULL,
  intensity_pmf = NULL,
  production = NULL
)
```

Arguments

sf	A sf object
keyTime	character. Name of the column for timeline. Used to check length of vectors in column vector object
key	character. Name of the column which is going to be created
FUN	A function applied on the sf object. See lapply functions for further details assuming X = 1:nrow(sf), that is sf[[key]] <- lapply(1:nrow(sf), FUN, ...).
timeline	Vector of time units (e.g. days) covering all the function
emission	Vector or Matrix given the quantity of emission per time unit per spatial unit. Length of vector equal the length of the 'timeline' vector (time unit matching). Size of the matrix, 'n,m', is such as the number of row equal the number of sources in 'sf' object, and the number of column equals the length of the 'timeline' vector.
density	Scalar or Vector (with length equal to the number of sources in 'sf' object) given the density of the source(s) (e.g. number of plant by squared meter)
intensity	Vector or Matrix given the quantity of emission per time unit per source. Length of vector equal the length of the 'timeline' vector (time unit matching). Size of the matrix, 'n,m', is such as the number of row equal the number of sources in 'sf' object, and the number of column equals the length of the 'timeline'.
intensity_pmf	Vector or Matrix given distribution of emission along time (given a probability mass function with time) . Length of vector equal the length of the 'timeline' vector (time unit matching). Size of the matrix, 'n,m', is such as the number of row equal the number of sources in 'sf' object, and the number of column equals the length of the 'timeline'.
production	Scalar or Vector (with length equal to the number of sources in 'sf' object) total emission for one source (e.g. total number of grains by plant)

Details

This function is a wrapper of with and lapply function and is like this: sf[["key"]] <- with(sf, lapply(1:nrow(sf), FUN, ...))

So, all column of sf can be called in FUN

Value

A matrix indexed by sources ID (in rows) and by time (in columns) whose rows give the values of intensity emission (number of grains) for every source.

brk_exposure *Compute exposure for a 'RasterStack' class object from package 'raster'*

Description

Compute exposure for a 'RasterStack' class object from package 'raster'

Usage

```
brk_exposure(
  RasterStack_dispersal,
  sf,
  key,
  keyTime,
  loss,
  beta,
  nbr_cores,
  quiet,
  unit
)
```

Arguments

RasterStack_dispersal	RasterStack. An object of classRasterStack (see package raster for details).
sf	sf. And object of class 'sf' on which exposure is computed from the previous list of raster by patch 'RasterStack_dispersal'. See sf for details.
key	name of the column in 'sf' object providing emission amount
keyTime	name o the column of sf for time
loss	numeric. scalar or vector (of the same length as the number the timeline include is argument sf) to apply a loss on exposure cells.
beta	numeric. toxic adherence parameter between 0 and 1 (default = 0.4).
nbr_cores	integer. Set the number of cores to used for parallel computing.
quiet	boolean. Set 'TRUE' to remove progress bar.
unit	default is meter "m". but should be more generic: "any".

Details

Local intensity depends of beta and alpha parameters. Beta represents the toxic adherence between [0,1]. Alpha represents a list of parameters of the lost of toxic particules due to covariates (precipitation). There are two configurations to integrate the loss in the function : (i) simulating covariate (simulate=TRUE) or (ii) uploading covariate (simulate=FALSE). The covariate is linked to the loss by a linear regression with paramaters minalpha, maxalpha, covariate_threshold.

brk_exposureMatch *Add raster value to element of sf object*

Description

Add the raster value(s) of a Raster* object to element of sf object.

Usage

```
brk_exposureMatch(
  stackRaster_exposure,
  sf,
  stackTimeline,
  keyTime = "TIMELINE",
  key = "EXPOSURE"
)
```

Arguments

stackRaster_exposure	The Raster* object
sf	the sf object
stackTimeline	sequence with matching to elements of RasterStack (length of stackTimeline must be the same as length of a list of stackRaster_exposure)
keyTime	name of the column to match exposure timeline from stackRaster_exposure
key	name of the new column

brk_FDtoDF *Convert list.column data.frame into scalar.column data.frame*

Description

Convert data.frame with 2 column.list into data.frame with only column.scalar

Usage

```
brk_FDtoDF_(sf, key1, key2, id = NULL)

brk_FDtoDF(sf, key1, key2, id = NULL, keep = NULL)

brk_FDtoDF_STICK(sf, key1, keep)
```

Arguments

sf	sf. An object of class sf
key1	character string. The name of the column to select
key2	icharacter string. The name of the column to select
id	name of the replicate for the id. As to be of the same length as the number of row of the sf object
keep	vector of column name to keep

---

brk_findIndexFD *find index*

Description

find index

Usage

```
brk_findIndexFD(sf, key, value)
```

Arguments

sf	An object of class sf
key	character string. name of the column to select
value	value of the element to return index from the column defined by key

Value

vector if not all index are equal. scalar if all equal.

---

brk_newPoints *Simulate new points on a specific sf object*

Description

Simulate new points on a specific sf object. See st_sample for details.

Usage

```
brk_newPoints(sf, size = 100)
```

Arguments

sf	object of class sf or sfc
size	sample size(s) requested; either total size, or a numeric vector with sample sizes for each feature geometry. When sampling polygons, the returned sampling size may differ from the requested size, as the bounding box is sampled, and sampled points intersecting the polygon are returned.

brk_rbindLStoDF *Combine list of data.frame by Rows*

Description

Function used to filter functional data in sf objects.

Usage

```
brk_rbindLStoDF(ls, id = NULL)
```

Arguments

ls	A list of data.frame
id	id to provide to each data.frame. Must be the length of the list

Value

Return a data.frame

brk_sampling *Sampling based on distribution provided by Raster* object over a sf object*

Description

Sampling based on distribution provided by Raster* object (does not have to be summed to one) over an sf object used as mask

Usage

```
brk_sampling(rasterStack, sf, sizeSite = 1)
```

Arguments

rasterStack	Raster* object (RasterLayer or RasterStack)
sf	An sf object
sizeSite	scalar or vector of the number of individual to sample per site. Default is 1. If vector, as to be the length of Raster* object

brk_timeline *Add sequence to element of a sf object*

Description

Add column with sequence for each row to a sf object

Usage

```
brk_timeline(sf, key, from, to, by)
```

Arguments

sf	object of class sf or sfc
key	a character string used as name of the new column
from	the starting value of the sequence. Of length 1 unless just from is supplied as an unnamed argument.
to	the end (maximal) value of the sequence. Of length 1 unless just from is supplied as an unnamed argument.
by	number: increment of the sequence.

brk_toxFun *Functions for Toxicokinetic-Toxidynamic (TKTD) models*

Description

Functions for Toxicokinetic-Toxidynamic (TKTD) models

ODE solver applied to IT-GUTS model

ODE solver applied to SD-GUTS model

Usage

```
brk_toxFun_damage1(exposure, kin, kout)

brk_toxFun_survival1(damage, alpha1, alpha2, alpha3)

brk_toxFun_survival2(damage, LC50, slope)

brk_survIT(time, Cw, listParameters)

brk_survSD(time, Cw, listParameters)
```

Arguments

exposure	exposure level of individual
kin	parameter describing the intake rate of the element
kout	parameter describing the excretion rate of the element
damage	damage level on which the survival model is going to be applied
alpha1	parameter describing the natural background death mortality rate
alpha2	parameter describing the killing rate of the element on individual
alpha3	exponential parameter describing the killing rate of the element on individual
LC50	parameter describing the lethal concentration for 50% of the population
slope	parameter describing the slope of the curve
time	vector of time of the exposure profile
Cw	vector of concentration of the exposure profile
listParameters	A list of parameter for the SD model

---

create_pollen_sources *Pollen sources emission simulation*

Description

Simulate pollen sources emission for maize crop. The proportion of plants emitting pollen per day (during 12 days) was observed by Frédérique Angevin (Angevin et al. 2008).

Usage

```
create_pollen_sources(
  sf,
  timeline = 1:61,
  density = runif(1, 7, 11),
  pollen = rgamma(1, shape = 1.6, scale = 1/(2 * 10^-7))
)

create.pollen.sources()
```

Arguments

sf	A sf object defining sources fields
timeline	Vector of interger. Time units (e.g. days) including the pollen emission (for simulation, default is time.max). Minimal value is a vector of 12 days.
density	Plant density (number of plant by squared meter)
pollen	Pollen production (number of grains by plant)

Value

A matrix indexed by sources ID (in rows) and by time (in columns) whose rows give the values of pollen emission (number of grains) for every source.

data_brk *Data set included in the package*

Description

Data set included in the package

Usage

```
data(maize_65)

data(sfMaize65)

data(maize.emitted_pollen)

data(maize.proportion_pollen)

data(Hofmann_2009)

data(Lang_2004)

data(Precipitation)

data(temperatureGermany)

data(df_precipitation)
```

Arguments

maize_65	A SpatialPolygonsDataFrame of class SpatialPolygonsDataFrame defining a patchy landscape
sfMaize65	A set of MULTIPOLYGON of class sf defining a patchy landscape
maize.emitted_pollen	A data.frame of pollen emission
maize.proportion_pollen	A data.frame of pollen proportion
Hofmann_2009	A data.frame of pollen emission
Lang_2004	A data.frame of pollen emission
Precipitation	data.frame of daily Precipitation from 01/01/2003 to 31/12/2013
temperatureGermany	A data.frame of temperature over a year in south and north of germany
df_precipitation	A data.frame of daily Precipitation from 01/01/2003 to 31/12/2013

GetInternProjection *Get the internal working projection PROJ.4*

Description

Will print and return the internal projection of briskaR package

Usage

```
briskaRGetInternProjection()
```

is_square_sf *Test if an sf is a square*

Description

Test if an sf is a square, with tolerance. Default is 5

Usage

```
is_square_sf(.sf, tolerance = 0.05)
```

Arguments

.sf	and object of class sf
tolerance	tolerance rate between both square side length

LAMBERT_93 *LAMBERT_93*

Description

SIG projection Lambert_93 references "+init=epsg:2154" PROJ.4

Usage

```
LAMBERT_93
```

Format

An object of class character of length 1.

loadIndividuals *Wrapper function loadIndividuals*

Description

Wrapper function to create an individuals object using sf or SpatialPoints and data.frame.

The SpatialPoints object and the data.frame have to contain the same number of coordinates and rows.

Usage

```
loadIndividuals(sf, data, timeline)
```

Arguments

sf	a sf object
data	a data.frame containing individuals attributes. Rows numbers as individuals ID, columns names as dob (date of birth) | life_duration | toxic_threshold
timeline	Vector of the time line

Value

an Individuals object

loadLandscape *Wrapper function : loadLandscape*

Description

Wrapper function to create a Landscape object using SpatialPolygons and dataframe. The SpatialPolygons object and the data.frame have to contain the same number of polygons and row (row ID is polygons ID).

Usage

```
loadLandscape(sp, data)
```

Arguments

sp	a SpatialPolygons object designing the landscape
data	a data.frame containing fields (polygons) information. Row names as fields ID, column names as sources | neutral | receptors (for a given field, the value is 1 for the type of the field (source or neutral or receptor), otherwise 0).

Value

A SpatialPolygonsDataFrame object

Examples

```
data(maize_65)
maize_data <- maize_65@data
maize_sp_only <- maize_65 ; maize_sp_only@data = data.frame(remove = rep(0,nrow(maize_65@data)))
load_landscape <- loadLandscape(maize_sp_only, maize_data)
```

---

loadLandscapeSIG *Create a Landscape object from SIG shapefile file*

Description

Create a Landscape object from SIG shapefile. Shapefile has to contain a SpatialPolygonsDataFrame. Data in the data frame contain fields (polygons) information. Row names as fields ID, cols names as sources | neutral | receptors (for a given field, the value is 1 for the type of the field (source or neutral or receptor), otherwise 0).

Usage

```
loadLandscapeSIG(dsn, layer, format = TRUE)
```

Arguments

dsn	folder path to the source files
layer	file name without extension
format	only load data needed Landscpae-class (default TRUE)

Value

A SpatialPolygonsDataFrame object

Examples

```
## Not run:
land <- loadLandscapeSIG("/path/to/directory/","fileName")
plot(land)

## End(Not run)
```

`loss_precipitation` *Loss precipitation function from a set of function implemented in earlier version of briskaR*

Description

Loss precipitation function from a set of function implemented in earlier version of briskaR

Usage

```
loss_precipitation(
  starttime = "16/07",
  endtime = "14/07",
 alpha = list(minalpha = 0.1, maxalpha = 0.95, covariate_threshold = 30, simulate =
    TRUE, covariate = NULL)
)
```

Arguments

`starttime`	data of begining of simultation
`endtime`	data of end of simultation
`alpha`	list of parameter

`saveIntoFile` *Save Particles Dispersion 3D Array to tiff file*

Description

Save into tiff file particles dispersion 3D array from toxicIntensity. The output is a RasterStack with a layer per time unit with projection set to CRS="+proj=longlat +datum=WGS84"

Usage

```
saveIntoFile(
  objectL,
  objectT,
  filename = "ParticlesDispersion.tif",
  format = "GTiff",
  overwrite = TRUE
)
```

Arguments

objectL	a Landscape object
objectT	a 3D array particles dispersion indexed by time (output from toxicIntensity)
filename	output file name (default "ParticlesDispersion.tif")
format	output format (default=GTiff)
overwrite	if TRUE overwrite file (default TRUE)

Value

a RasterStack object

Examples

```
## Not run:
data(maize_65)
ti <- toxicIntensity(maize.landscape,maize.emitted_pollen)
saveIntoFile(maize.landscape,ti,filename="ParticlesDispersion.tiff",format="GTiff",overwrite=T)

## End(Not run)
```

---

simul.precipitation *Simulate precipitation between two dates*

Description

Will evaluate parameters from data and simulate precipitation between the two dates.

Usage

```
simul.precipitation(starttime = "15/07", endtime = "15/09", data = NULL)
```

Arguments

starttime	*string*. Date shape: "mm/yy"
endtime	*string*. Date shape: "mm/yy"
data	*data.frame*. default is NULL

Value

an array of length of the period between the two dates included

Author(s)

Jean-Francois Rey

`simulateIndividuals` *Wrapper function SimulateIndividuals*

Description

This function simulates individuals as an Individuals object.

Will simulate `n` individuals in receptors fields of a landscape.

Usage

```
simulateIndividuals(
  sf,
  size = 100,
  timeline = 1:61,
  dob,
  life_duration,
  toxic_threshold
)
```

Arguments

`sf`	A sf object
`size`	Number of individuals to simulate
`timeline`	Vector of the time line
`dob`	A vector for the Date Of Birth of each individual between `min` and `max` of the `timeline`.
`life_duration`	A vector for the life duration of each individual
`toxic_threshold`	A vector for the internal toxic threshold value leading to death for each individual

Details

The Individuals object output includes for each individual the coordinates, the date of birth, the life duration, the toxic threshold and

simulateInitialPartition

simulateInitialPartition Method

Description

This function creates an object SpatialPolygonsDataFrame and simulates a landscape with neutral and source fields.

Usage

```
simulateInitialPartition(
  n = 500,
  prop = 0.4,
  range = 10,
  xmin = 0,
  xmax = 5000,
  ymin = 0,
  ymax = 5000
)
```

Arguments

n	Numeric, numbers of cells
prop	Numeric [0,1] toxic cells proportion
range	Aggregation parameter (range of the spatial Exponential covariance of Gaussian process)
xmin	x-axis left coordinates in space unit (see projections_briskaR)
xmax	x-axis right coordinates in space unit (see projections_briskaR)
ymin	y-axis bottom coordinates in space unit (see projections_briskaR)
ymax	y-axis top coordinates in space unit (see projections_briskaR)

Details

In the function the first step is a binomial point process to simulate a SpatialPointsDataFrame with sources and neutral marks, which depends on the aggregation parameter. The second step of the function is the Voronoi tesselation from the simulated points and returns a SpatialPolygons-DataFrame.

Value

An SpatialPolygonsDataFrame object with n fields, prop pourcentage of toxic fields of size (xmin,xmax) (ymin,ymax)

Examples

```
## Not run:
# Simulate a 5000m x 5000m landscape with 500 cells (e.g. fields)
# whose 40% (200 cells) are sources.
# The projection by default is Lambert93 projection.
land <- simulateInitialPartition(n=500,prop=0.4,range=10,xmin=0,xmax=5000,ymin=0,ymax=5000)
plot(land)

## End(Not run)
```

`simulateLandscape` *Simulate a new landscape*

Description

Create an object of class SpatialPolygonsDataFrame. Simulate a landscape with neutral and source fields and receptors margins.

Usage

```
simulateLandscape(
  n = 500,
  prop = 0.4,
  range = 10,
  xmin = 0,
  xmax = 5000,
  ymin = 0,
  ymax = 5000,
  border_size = 200,
  prob = runif(1, 0.1, 0.9),
  mean_thickness = runif(1, 2, 20),
  v_thickness = 50
)
```

Arguments

`n`	Numeric, numbers of fields
`prop`	Numeric [0,1] toxic fields proportion
`range`	aggregation parameter (range of the spatial exponential covariance of gaussian process) in meters.
`xmin`	x-axis left coordinates
`xmax`	x-axis right coordinates
`ymin`	y-axis bottom coordinates
`ymax`	y-axis top coordinates

border_size	A numeric, bbox margin
prob	Probability to inflate a filed margin
mean_thickness	Margin width expectation
v_thickness	Margin width variance

Details

Execute both simulateInitialPartition and simulateThickMargins functions.

Value

A SpatialPolygonsDataFrame object with n fields, prop pourcentage of toxic fields.

See Also

simulateInitialPartition and simulateThickMargins

Examples

```
## Not run:
land <- simulateLandscape(n=100, prop=0.4, range=10,
xmin=0, xmax=1000, ymin=0, ymax=1000, border_size=100,
prob=runif(1,0.1,0.9), mean_thickness=runif(1,2,20),
v_thickness=50)
plot(land)
## End(Not run)
```

---

simulateThickMargins *Simulate thick margin to a landscape*

Description

Simulate thick margins as receptors in a landscape.

Usage

```
simulateThickMargins(
  objectL,
  border_size = 200,
  prob = runif(1, 0.1, 0.9),
  mean_thickness = runif(1, 2, 20),
  v_thickness = 50
)
```

Arguments

objectL	sf, sp or Landscape (earlier version of briskaR).
border_size	A numeric, bbox margin
prob	Probability to inflate a margin
mean_thickness	Margin width expectation in meter
v_thickness	Margin width variance in meter

Details

Margin width use a Gamma distribution with shape and scale parameters based on thickness mean and variance.

Value

A SpatialPolygonsDataFrame object

See Also

simulateInitialPartition and simulateLandscape

Examples

```
## Not run:
data(maize_65)
plot(maize.landscape)
landscape.margin <- simulateThickMargins(maize.landscape)
plot(landscape.margin)
## End(Not run)
```

---

st_multibuffer *Simulate thick margin to a landscape*

Description

Add buffer around each objects of a sf file

Usage

```
st_multibuffer(
  sf,
  dist = 50,
  nQuadSegs = 30,
  endCapStyle = "ROUND",
  joinStyle = "ROUND",
  mitreLimit = 1
)
```

Arguments

sf	object of class sfg, sfg or sf
dist	numeric; buffer distance for all, or for each of the elements in x; in case dist is a units object, it should be convertible to arc_degree if x has geographic coordinates, and to st_crs(x)$units otherwise
nQuadSegs	integer; number of segments per quadrant (fourth of a circle), for all or per-feature
endCapStyle	character; style of line ends, one of 'ROUND', 'FLAT', 'SQUARE'
joinStyle	character; style of line joins, one of 'ROUND', 'MITRE', 'BEVEL'
mitreLimit	numeric; limit of extension for a join if joinStyle 'MITRE' is used (default 1.0, minimum 0.0)

Details

see package st_buffer for details

---

st_squared_geometry *Add squared frame polygon*

Description

Return a square frame surrounding a list of sf

Usage

```
st_squared_geometry(list_sf, buffer_dist = NULL)
```

Arguments

list_sf	A list of objects of class sf
buffer_dist	numeric; buffer distance for all, or for each of the elements in x; in case dist is a units object, it should be convertible to arc_degree if x has geographic coordinates, and to st_crs(x)$units otherwise. See function 'sf_buffer' from package 'sf' for details

`toxicIntensity` *toxicIntendity function wrapping dispersal and exposure*

Description

toxicIntendity function wrapping dispersal and exposure

Usage

```
toxicIntensity(
  object,
  sf,
  size_raster = 2^10,
  tolerance_square = 0.1,
  kernel = "NIG",
  kernel.options = list(a1 = 0.2073, a2 = 0.2073, b1 = 0.3971, b2 = 0.3971, b3 =
    0.0649, theta = 0),
  loss = NULL,
  beta = 0.4,
  nbr_cores = 1,
  squared_frame = NULL,
  quiet = FALSE
)
```

Arguments

`object`	sf or SpatialPolygonsDataFrame. A simple feature of class sf or SpatialPolygonsDataFrame
`sf`	sf. And object of class 'sf' on which exposure is computed from the previous list of raster by patch 'RasterStack_dispersal'. See sf for details.
`size_raster`	integer. Raster size (default = 2^10)
`tolerance_square`	numeric. Tolerance rate to test if an sf set is squared
`kernel`	string. Dispersion kernel, function name (default = NIG)
`kernel.options`	list. Parameters list for the kernel function
`loss`	numeric. Numeric vector to applied a loss on exposure cells.
`beta`	numeric. toxic adherence parameter between 0 and 1 (default = 0.4).
`nbr_cores`	integer. Parameters for parallel computing: the number of cores to use, i.e. at most how many child processes will be run simultaneously. Default is 1 (non parallel).
`squared_frame`	sf. Select the sf to be considered as frame to rasterized. Default is 'NULL', and 'object' is used.
`quiet`	boolean. Set 'TRUE' to remove progress bar.

Details

The dispersal of contaminants is implemented by rastering the landscape and by computing the convolution between sources emissions and a dispersal kernel.

The dispersion kernel by default is Normal Inverse Gaussian kernel ("NIG" function). Currently, two others are implemented "geometric" (with parameter `a`) and "2Dt" kernels (with parameters `a`, `b`, `c1`, `c2`).

Local intensity depends of `beta` and `alpha` parameters. Beta represents the toxic adherence between [0,1]. Alpha represents a list of parameters of the lost of toxic particules due to covariates (precipitation). There are two configurations to integrate the loss in the function : (i) simulating covariate (simulate=TRUE) or (ii) uploading covariate (simulate=FALSE). The covariate is linked to the loss by a linear regression with paramaters minalpha, maxalpha, covariate_threshold.

Index