Package 'secrlinear'

May 9, 2026

Type Package

Title Spatially Explicit Capture-Recapture for Linear Habitats

Version 1.2.5

Depends R (>= 3.5.0), secr (>= 3.0.1)

Imports MASS, methods, sp, igraph, sf

Suggests knitr, rmarkdown, secrdesign, spatstat (>= 3.0-2), spatstat.geom, spatstat.linnet, testthat

VignetteBuilder knitr

Date 2026-01-09

Maintainer Murray Efford <murray.efford@otago.ac.nz>

Description Tools for spatially explicit capture-recapture analysis of animal populations in linear habitats, extending package 'secr'.

License GPL (>= 2)

LazyData yes

URL https://www.otago.ac.nz/density/, https://github.com/MurrayEfford/secrlinear/

NeedsCompilation no

Author Murray Efford [aut, cre] (ORCID: <https://orcid.org/0000-0001-5231-5184>)

Repository CRAN

Date/Publication 2026-01-09 06:10:02 UTC

Contents

`secrlinear-package` *Spatially Explicit Capture–Recapture for Linear Habitats*

Description

An **secr** add-on package for linear habitat models.

Details

Package:	secrlinear
Type:	Package
Version:	1.2.5
Date:	2026-01-09
License:	GNU General Public License Version 2 or later

The important functions in **secrlinear** are:

`read.linearmask`	import and discretize a linear habitat map (route)
`networkdistance`	compute network distances between detectors and points using a linear habitat mask
`sim.linearpopn`	simulate population along linear mask
`make.line`	place detectors along (part of) a linear route

Other useful functions are:

`checkmoves`	check capthist object for extreme movements
`showpath`	interactive examination of network distances
`clipmask`	drop mask points outside buffer distance
`rbind.linearmask`	combine two linear masks
`subset.linearmask`	select part of a linear mask

asgraph	convert linear mask to igraph
snapPointsToLinearMask	closest point on graph
make.sldf	convert coordinates to SpatialLinesDataFrame

Documentation is provided in a vignette secrlinear-vignette.pdf and in the pdf version of the help pages.

The package draws on the packages **sp** (Pebesma and Bivand 2005) and **igraph** (Csardi and Nepusz 2006).

In order to fit a linear-habitat model with the **secr** function `secr.fit`:

1. specify a linear mask for the 'mask' argument
2. specify `details = list(userdist = networkdistance)` so that `secr.fit` uses network distances rather than Euclidean distances.

An example dataset arvicola is drawn from trapping of water voles *Arvicola amphibius* in June 1984 along the River Glyme in the U.K. – the relevant linear mask is glymemask.

A more complex linear network (Silverstream) is provided as an ESRI shapefile.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

References

Csardi, G. and Nepusz, T. (2006) The igraph software package for complex network research. *InterJournal, Complex Systems* **1695**. https://igraph.org/.

Pebesma, E.J. and Bivand, R. S. (2005) Classes and methods for spatial data in R. *R News* **5**(**2**), https://cran.r-project.org/doc/Rnews/.

See Also

addedges, asgraph, checkmoves, clipmask, deleteedges, linearmask, linearpopn, make.line, make.sldf, networkdistance, plot.linearmask, rbind.linearmask , read.linearmask, showpath, showedges, sim.linearpopn, snapPointsToLinearMask, subset.linearmask

Examples

```
## Water voles in June 1984 on the R. Glyme in Oxfordshire, UK
## capture and trap location files are exactly as for a 2-D analysis

inputdir <- system.file("extdata", package = "secrlinear")
arvicola <- read.capthist(captfile = paste0(inputdir, "/Jun84capt.txt"),
    trapfile = paste0(inputdir, "/glymetrap.txt"), detector = "multi",
    covname = "sex")

## Import map of linear habitat
## -- from text file of x-y coordinates
```

```
glymemask <- read.linearmask(file = paste0(inputdir, "/glymemap.txt"), spacing = 4)
## -- from a previously constructed SpatialLinesDataFrame
SLDF <- as(sf::st_read(paste0(inputdir,"/glymemap.shp")), 'Spatial')
glymemask <- read.linearmask(data = SLDF, spacing = 4)

## display the mask and capture data
plot (glymemask)
plot(arvicola, add = TRUE, tracks = TRUE)
plot(traps(arvicola), add = TRUE)

## fit model, estimate density
linearfit <- secr.fit(arvicola, mask = glymemask, trace = FALSE,
    details = list(userdist = networkdistance))
predict(linearfit)

## NOTE : the unit of density (D) is animals / km
```

`Arvicola` *Water Vole Capture Dataset*

Description

Data from a trapping study of water voles (*Arvicola amphibius*) along a river in Oxfordshire, U.K.

Usage

```
arvicola
```

Format

secr capthist object

Details

Water voles were trapped monthly from May 1984 to May 1985 along 0.9 km of the River Glyme near Woodstock in Oxfordshire, U.K. (Efford 1985). Two sheet-aluminium traps were set at stations 20 m apart along one bank and checked morning and evening for 3 days. Traps were baited with slices of carrot and provisioned with bedding and additional carrot. Voles were marked with individual colour-coded ear tags. The dataset provided is from June 1984. This was early in the breeding season and most voles were overwintered adults; only 3 were young-of-the-year, and these were omitted.

Raw data files "Jun84capt.txt" and "glymetrap.txt" are provided in the 'extdata' folder of the **secrlinear** installation. The vignette `../doc/secrlinear-vignette.pdf` shows how to import the raw data.

The data comprise detections of 26 voles at 44 stations on 6 occasions. The two traps at each station were notionally labelled 'A' and 'B', but captures were recorded only by the station at which they occurred: captures were assigned label 'A' or 'B' effectively at random. Each trap could catch a

single adult vole, but trap saturation was low (maximum 20.5% of traps caught a vole). No voles died in traps in June 1984.

Water voles in the U.K. restrict their activity to waterways and their immediate banks, except for some populations in more extensive (2-dimensional) wetlands. It is therefore natural to treat their habitat as linear in a spatially explicit capture–recapture model of these data. A suitable linear habitat mask is provided in the accompanying dataset glymemask.

See `../doc/secrlinear-vignette.pdf` for more analysis of this dataset.

Source

M. G. Efford unpublished data.

References

Efford, M. G. (1985) *The structure and dynamics of water vole populations*. D.Phil thesis, University of Oxford.

See Also

glymemask

Examples

```
head(traps(arvicola))

## for speed, pre-compute distance matrix
userd <- networkdistance (traps(arvicola), glymemask, glymemask)
## fit model
glymefit <- secr.fit(arvicola, mask = glymemask, trace = FALSE,
                     details = list(userdist = userd))
## estimates of 'real' parameters
predict(glymefit)

## Not run:
summary(arvicola)

tmp <- secr.test(glymefit, nsim = 1000)
tmp
plot(tmp)

## More voles were caught only once than is predicted by the model.
## This is probably due to within-population variation in movement or
## capture probability.


## End(Not run)
```

checkmoves *Check Movements*

Description

This function reports problems indicated by extreme movements in a capthist object. At present it is configured for distances measured along a linearmask only.

Usage

```
checkmoves (CH, accept = c(0, 1000), userdist, mask, showall = TRUE, silent = FALSE)
```

Arguments

CH	single-session secr capthist object
accept	range of accepted movements in metres; may include Inf
userdist	matrix of distances between all traps(CH) (optional)
mask	linearmask object (optional)
showall	logical; if TRUE the output dataframe includes all detections of each individual with at least one extreme movement
silent	logical; if TRUE then console output is suppressed

Details

This function works with Euclidean distances (the default if neither mask nor userdist is specified), non-Euclidean network distances implied by a linear habitat mask, or any distances presented in userdist.

Linear habitat masks are prone to breaks (missing edges in the graph representation). If an animal is 'recaptured' on both sides of a break then the network distance for that 'movement' will be infinite.

checkmoves displays the message 'All OK' if there are no out-of-range movements, and otherwise displays the number of individuals with out-of-range movements.

Value

Invisibly returns a list with components

badmoves	logical vector with one element for each animal (i.e. each row in CH)
CH	subset of the input CH containing the faulty detection histories
df	dataframe detailing the faulty detections

See Also

linearmask

Examples

```
out <- checkmoves(captdata, accept = c(0,100))
out
```

`clipmask` *Drop Points Distant From Detectors*

Description

`clipmask` may be used to drop superfluous points from a mask (those further than `buffer` from any detector).

Usage

```
clipmask(mask, traps, buffer = 100, clipvert = FALSE)
```

Arguments

`traps`	'traps' object from **secr**
`mask`	'linearmask' object
`buffer`	network distance for retaining mask points (m)
`clipvert`	logical; if TRUE the underlying lines are also clipped

Details

None yet.

Value

'linearmask' object retaining only points within `buffer` m of any trap. By default, the 'SLDF' attribute is unchanged.

`clipvert = TRUE` causes the attribute "SLDF" (a SpatialLinesDataFrame) to be clipped to within spacing/2 of detectors, using functions from **sf** (Pebesma 2018).

References

Pebesma, E. (2018) Simple features for R: standardized support for spatial vector data. *The R Journal* **10(1),** 439–446. doi:10.32614/RJ2018009

See Also

`make.line`

Examples

```
x <- seq(0, 4*pi, length = 200)
xy <- data.frame(x = x*100, y = sin(x)*300)
mask <- read.linearmask(data = xy, spacing = 20)

## clipmask
trps <- make.line(mask, n = 15, startbuffer = 1000, by = 30)
newmask <- clipmask(mask, trps, buffer = 200)
plot(newmask)
plot(trps, add = TRUE)

newmask <- clipmask(mask, trps, buffer = 200, clipvert = TRUE)
plot(newmask)
plot(trps, add = TRUE)
```

---

edges *Interactive Edge Edit*

Description

A 'skip' is defined here as a graph edge between two input lines. Many skips are legitimate, but some 'shortcut' skips are artefacts of the simple algorithm used by asgraph to construct the graph topology. These are better removed. These functions allow skips in a linearmask to be visualised and edited.

addedges may be used to bridge artefactual gaps in a network, or to correct editing mistakes. The edge weight of a new edge is its Euclidean length.

Usage

```
showedges(mask, plt = TRUE, add = FALSE, type = c('all', 'noskips', 'skips'),
                        lengths = c(0,Inf), ...)

replot(mask, xlim = NULL, ylim = NULL, ...)

addedges(mask, replot = TRUE, ...)

deleteedges(mask, replot = TRUE, ...)

cleanskips(mask)
```

Arguments

mask	linearmask object with 'graph' attribute
lengths	numeric vector of minimum and maximum edge length (weight)
plt	logical; if TRUE the edges are plotted

add	logical; if TRUE the edges will be added to an existing plot
type	character value for subset of edges to show
xlim	vector of min and max x-values for new plot
ylim	vector of min and max y-values for new plot
replot	logical; if TRUE the plot will be re-done using replot
...	other arguments (see details)

Details

showedges is a general-purpose function for displaying or extracting edge information from the graph attribute of a linear mask.

cleanskips removes any between-line 'skips' that are longer than the shortest 'skip'.

Other functions here are used to manually edit a mask graph.

replot is used to zoom in on part of a linear mask (often to inspect 'skips').

The dots argument ... is passed by replot to plot.linearmask and by addedges and deleteedges to showedges (if replot = TRUE).

Value

showedges returns a dataframe with one row for each edge, and columns –

start	index of start vertex
finish	index of finish vertex
line1	line number of start
line2	line number of finish
x1	geographic coordinates of start
y1	geographic coordinates of start
x2	geographic coordinates of finish
y2	geographic coordinates of finish
weight	"weight" attribute of edge (its length in metres)

The dataframe is returned invisibly if plt = TRUE.

cleanskips returns a linearmask object.

replot returns NULL.

addedges and deleteedges invisibly returns a linear mask object with updated graph attribute.

See Also

linearmask, getLineID

Examples

```
## Not run:
inputdir <- system.file("extdata", package = "secrlinear")
tempmask <- read.linearmask(file = paste0(inputdir, "/silverstream.shp", spacing = 50)

## show all edges
plot(tempmask, linecol = 'white')
tmp <- showedges(tempmask, add = TRUE)

## select a rectangular area to zoom in
replot(tempmask)

## click on the vertices of one or more edges to delete or add
tempmask <- deleteedges(tempmask)
tempmask <- addedges(tempmask)


## End(Not run)
```

`glymemask` *Linear Mask for Water Vole Dataset*

Description

Habitat geometry for a trapping study of water voles (*Arvicola amphibius*) along a river in Oxfordshire, U.K., used in conjunction with the capture dataset arvicola.

Usage

```
glymemask
```

Format

A linearmask object (see `read.linearmask`)

Details

Water voles were trapped monthly from May 1984 to May 1985 along 0.9 km of the River Glyme in Oxfordshire, U.K. (Efford 1985).

The raw data are provided both as a text file and as a shapefile in the 'extdata' folder of the **secrlinear** installation. The vignette `../doc/secrlinear-vignette.pdf` shows how to import the data, which are also provided here as an **secr** object.

Source

M. G. Efford unpublished data.

References

Efford, M. G. (1985) *The structure and dynamics of water vole populations*. D.Phil thesis, University of Oxford.

See Also

arvicola

Examples

```
plot(glymemask)
summary(glymemask)

## add a covariate, the distance downstream from the first mask point
downstrm <- networkdistance(glymemask, glymemask[1,], glymemask)[,1]
covariates(glymemask)<- data.frame(downstream = downstrm)
```

---

linearkd *Kernel Density on Linear Network*

Description

This function applies the kernel density methods of McSwiggan et al. (2016) as implemented in **spatstat** (Baddeley et al. 2015). The default method solves the heat equation McSwiggan et al. (2016).

Usage

```
linearkd(X, linmask, sigma, which = NULL, ...)
```

Arguments

X	2-column matrix of coordinates
linmask	linear habitat mask
sigma	numeric spatial scale of kernel
which	index vector to select subset of edges (optional)
...	other arguments passed to spatstat.linnet::density.lpp

Details

The density along the network of the points in X is saved as covariate 'density' of the linear mask.

Setting to FALSE the argument 'finespacing' of densityHeat.lpp speeds up estimation.

Value

A linear habitat mask identical to the input except for the additional covariate.

Warning

This function is still in development: there are details to resolve concerning terminal vertices etc.

References

Baddeley, A., Rubak, E., and Turner, R. 2015. Spatial Point Patterns: Methodology and Applications with R. Chapman and Hall/CRC Press, London. ISBN 9781482210200, https://www.routledge.com/Spatial-Point-Patterns-Methodology-and-Applications-with-R/Baddeley-Rubak-Turner/p/book/9781482210200/.

McSwiggan, G., Baddeley, A. and Nair, G. 2016. Kernel density estimation on a linear network. *Scandinavian Journal of Statistics* **44,** 324–345.

See Also

`read.linearmask`

Examples

```
# simulate some points
set.seed(123)
pop <- sim.linearpopn(glymemask, N = 40)

# restrict edges to overcome a glitch in this particular linearmask

tmp <- linearkd(X = pop, linmask = glymemask, sigma = 30,
    which = 1:325, finespacing = FALSE)

plot(tmp, cov = 'density', cex = 1.7)
plot (pop, add = TRUE, cex = 1.4)

# view covariates
head(covariates(tmp))
```

---

linearmask.object *Description of Linear Mask Objects*

Description

An object of class `linearmask` maps linear habitat, such as a river, in a way that is useful for spatially explicit capture–recapture analysis. The object contains two representations of the linear habitat: as features in a SpatialLinesDataFrame, and discretized as a set of equally spaced points, as in a mask object from **secr**.

Usually the object was created in a call to `read.linearmask`, which performs the discretization given linear input data and a chosen spacing.

A linearmask object does not explicitly include topological information i.e. the branching pattern of river networks. Instead, a network is constructed on-the-fly in `asgraph` by joining adjacent mask points (those within a certain small Euclidean distance). This results in 'shortcut' edges where lines approach at an acute angle (see the Examples for a demonstration).

A more complete, but also more cumbersome, solution for river networks might be to adapt the SpatialStreamNetwork object class of the **SSN** package (ver Hoef et al. 2014). Data input for **SSN** requires pre-processing in the proprietary software ArcGIS.

Value

The object itself is a dataframe of coordinates for the points comprising the mask, as for a 2-dimensional habitat mask (mask).

Additional information is stored as attributes (these usually remain hidden, but are essential except for 'graph'):

`spacingfactor`	The spacing factor determines which points in a linear mask are joined to form a linear network, as happens automatically when computing distances with `networkdistance`. Points are joined when their Euclidean separation is less than spacingfactor x spacing. The default value is 1.1.
`SLDF`	The SLDF attribute is a SpatialLinesDataFrame (defined in **sp**) that provides the underlying description of the linear habitat.
`graph`	**igraph** network. Optionally computed by `read.linearmask` and saved as attribute, or computed on the fly by `networkdistance`
.	
`covariates`	A required dataframe of covariate values, including at least the column 'LineID'

Other attributes are common to masks and linearmasks:

`spacing`	In this context, the length of habitat in metres that is associated with each mask point (segment length).
`meanSD`	A matrix with mean and standard deviation of coordinates.
`boundingbox`	In this context, a dataframe giving the coordinate ranges spanned by the SLDF. Differs from 2-D mask in not being 'buffered' by spacing/2.

Further attributes of 2-dimensional masks (area, polygon etc.) are not relevant but may be present. The object has class `c("linearmask", "mask", "data.frame")` or `c("list", "linearmask", "mask", "data.frame")` for a multi-session linear mask.

References

Ver Hoef, J. M., Peterson, E. E., Clifford, D. and Shah, R. (2014) **SSN** : an R package for spatial statistical modeling on stream networks. *Journal of Statistical Software* **56(3)**, 1–45.

See Also

`read.linearmask`, `mask`, `showpath`, `networkdistance`

Examples

```
## make a complex linear mask
x <- seq(0, 4*pi, length = 200)
xy <- data.frame(x = x*100, y = sin(x)*300)
xy2 <- data.frame(x = x*100, y = cos(x)*300)
test <- read.linearmask(data = xy, spacing = 10)
test2 <- read.linearmask(data = xy2, spacing = 10)
test3 <- rbind(test, test2)

## visualize the igraph network used by networkdistance()
tmp <- asgraph(test3)
if(require('igraph')) {
  i2 <- get.edges(tmp,1:ecount(tmp))
  plot(test3, linecol = 'white', col = 'grey')
  segments(test3[i2[,1],1], test3[i2[,1],2], test3[i2[,2],1],
      test3[i2[,2],2], lwd=2, col='blue')
}
```

---

linearpopn.object *Description of Linear Population Objects*

Description

An object of class linearpopn, usually created in a call to sim.linearpopn and used as input to sim.capthist.

Value

The object itself is a dataframe of coordinates for points along the mask, as for a 2-dimensional population (popn).

The attribute model2D is always "linear".

The object has class c("linearpopn", "popn", "data.frame").

See Also

plot.linearpopn, popn.

Examples

```
glymepop <- sim.linearpopn(glymemask, 30)
plot(glymepop, jitter=1)

CH <- sim.capthist(traps(arvicola), glymepop, detectpar = list(g0 = 0.4,
    sigma = 40), noccasions = 6, userdist = networkdistance)

plot(glymemask)
plot(CH, add = TRUE, tracks = TRUE)
```

make.line *Linear Detector Array*

Description

Build **secr** 'traps' object, positioning detectors along a predefined set of lines. `make.line` is in the same family as **secr** functions `make.grid` etc.

Usage

```
make.line(SLDF, n = 10, startbuffer = 0, by = 20, endbuffer = 0, cluster
= NULL,
type = c("fixedstart", "randomstart", "centred"), detector = "multi")
```

Arguments

`SLDF`	linearmask object or SpatialLinesDataFrame from **sp**
`n`	maximum number of detectors (or clusters of detectors) per line
`startbuffer`	distance of first detector from start of line(s)
`endbuffer`	minimum distance of last detector from end of line(s)
`by`	spacing between detectors (or clusters of detectors) (m)
`cluster`	numeric vector of within-cluster positions (m from start)
`type`	character
`detector`	character value for detector type - "single", "multi" etc.

Details

Detectors are placed independently on each line.

Value

An object of class `traps` comprising a data frame of x- and y-coordinates, the detector type ("single", "multi", or "proximity" etc.), and possibly other attributes. The SpatialLinesDataFrame is retained as attribute 'SLDF'.

See Also

`make.traps`

Examples

```
x <- seq(0, 4*pi, length = 200)
xy <- data.frame(x = x*100, y = sin(x)*300)
mask <- read.linearmask(data = xy, spacing = 20)

trps <- make.line(mask, n = 15, startbuffer = 1000, by = 50)
plot(mask)
plot(trps, add = TRUE)
```

`networkdistance` *Network Distances*

Description

Simple network computation of distance between points along linear landscape features (e.g. rivers).

Usage

```
networkdistance(xy1, xy2, geometry)
```

Arguments

`xy1`	2-column matrix or dataframe
`xy2`	2-column matrix or dataframe
`geometry`	'linearmask' object

Details

`networkdistance` computes the distance in metres between the points in `xy1` and the points in `xy2`. The `geometry` is a linearmask; if it is missing then it takes the value of `xy2`, which then of course must be a full linearmask. A network is taken from the 'graph' attribute of `geometry` or (if necessary) constructed from the linearmask 'on the fly', joining any points closer than (`spacingfactor` x `spacing(geometry)`). Points in `xy1` and `xy2` are first snapped to the nearest point in `geometry`. The computed distance is the sum of the graph's edge weights along the shortest path joining two points.

The commonest use of `networkdistance` is to calculate distances between detectors (traps) and points on a linear mask. In this case it is sufficient to provide two arguments only, as `xy2` serves as both destination and geometry.

`networkdistance` meets the requirements for a user-defined distance function (userdist) in **secr**. No parameter is estimated.

Use `showpath` to check network distances interactively.

Value

networkdistance –

given k detectors and m mask points, a k x m matrix of distances.

If called with no arguments networkdistance returns a zero-length character vector.

See Also

read.linearmask, showpath, asgraph

Examples

```
x <- seq(0, 4*pi, length = 200)
xy <- data.frame(x = x*100, y = sin(x)*300)
mask <- read.linearmask(data = xy, spacing = 20)
trps <- make.line(mask, n = 15, startbuffer = 1000, by = 30)

## networkdistance; geometry in 'mask'
d <- networkdistance (trps, mask, mask)
dim(d)
head(d)
```

---

plotmethods *Plot Methods for linearmask and linearpopn objects*

Description

Custom plotting.

Usage

```
## S3 method for class 'linearpopn'
plot(x, ..., jitter = 0, plotline = TRUE)

## S3 method for class 'linearmask'
plot(x, ..., linecol = 'black', label = FALSE,
                laboffset = c(spacing(x), 0))
```

Arguments

x	linearpopn object from sim.linearpopn, or a linearmask object from read.linearmask
...	For plot.linearpopn: other arguments passed to plot.popn(may include add = TRUE). For plot.linearmask: other arguments passed to plot.mask (e.g., col, cex, add), to legend (pt.cex) or to the plot method for SpatialLines (lwd, lty)
jitter	numeric value for jittering

plotline	logical; if TRUE the mask line is overplotted in grey
linecol	line colour for linear habitat (see Color Specification in par)
label	logical; if TRUE each vertex is numbered
laboffset	offset of label from point (metres)

Details

The linear mask used for plotting a 'linearpopn' is the one saved as an attribute of the object.

The main plotting in plot.linearmask is done by plot.mask with dots = TRUE. See the help for plot.mask for details of options such as add. The lines of the SpatialLinesdataFrame are overplotted unless linecol = NA.

Jittering shifts points by a random uniform distance – ($\pm 0.5\times$ jitter) x mask spacing – on both axes. This can give a better impression of density when points coincide.

The option inplot.popn for plotting rectangular 'frame' is suppressed: do not attempt to pass this argument in ....

Value

plot.linearpopn does not return a value.

plot.linearmask invisibly returns legend details as for plot.mask.

See Also

sim.linearpopn, plot.mask, plot.popn

Examples

```
x <- seq(0, 4*pi, length = 200)
xy <- data.frame(x = x*100, y = sin(x)*300)
mask <- read.linearmask(data = xy, spacing = 10)
linpop <- sim.linearpopn(mask, 100)
plot(linpop, jitter = 2)

plot(mask)

## thicker band of grey points, dashed line
plot(mask, cex = 2, lty = 2)

## add a covariate, the distance downstream from the first mask point
downstrm <- networkdistance(glymemask, glymemask[1,], glymemask)[,1]
covariates(glymemask)<- data.frame(downstream = downstrm)

## point colour determined by a covariate
plot(glymemask, cex = 2, covariate = 'downstream', pt.cex = 2)

## point size determined by a covariate
plot(glymemask, cex = covariates(glymemask)$downstream/50, pch = 21)
```

`rbind.linearmask` *Combine linearmask Objects*

Description

Form a new linearmask object by combining the underlying vertices and drawing a new systematic sample.

Usage

```
## S3 method for class 'linearmask'
rbind(..., cleanskips = TRUE)
```

Arguments

`...`	one or more linearmask objects with the same spacing
`cleanskips`	logical; passed to internal function `make.linearmask`

Details

The **sp** function `rbind` is used to combine the 'SLDF' attributes of the inputs.

If the input objects have a 'graph' attribute then a new graph will be included in the output.

Value

A linearmask object

See Also

`read.linearmask`, linearmask, `subset.linearmask`

Examples

```
x <- seq(0, 4*pi, length = 200)
xy <- data.frame(x = x*100, y = sin(x)*300)
xy2 <- data.frame(x = x*100, y = cos(x)*300)
test <- read.linearmask(data = xy, spacing = 20)
test2 <- read.linearmask(data = xy2, spacing = 20)

plot(rbind(test, test2))
```

`read.linearmask` *Import Linear Habitat Mask*

Description

Construct a linearmask from line data in a text file, a polyline shapefile, a dataframe or a SpatialLinesDataFrame object.

Usage

```
read.linearmask(file = NULL, data = NULL, spacing = 10, spacingfactor =
1.5, graph = TRUE, cleanskips = TRUE,  ...)
```

Arguments

`file`	file name
`data`	data frame or SpatialLinesDataFrame
`spacing`	length of each discretized segment
`spacingfactor`	numeric criterion for joining mask points
`graph`	logical; if TRUE the **igraph** representation of the mask is saved as an attribute
`cleanskips`	logical; if TRUE then `cleanskips` is applied to the graph
`...`	other arguments passed to `read.table` for file input

Details

Only one of 'file' or 'data' should be specified. The file name should include the extension (e.g., "silverstream.shp").

Input from a text file, polyline shapefile or dataframe is first converted to a SpatialLinesDataFrame object. Points are spaced equally along each SpatialLine, starting at spacing / 2.

For dataframe input via `data` the coordinates are expected to be in columns 'x' and 'y'; if these names are missing, or input is from a file, the first two columns are used.

`spacingfactor` is used by `asgraph` to determine which points are joined when computing distances. It should be between 1 and 2.

Value

A 'linearmask' object, i.e. a 'mask' object in which each point represents a line segment of length `spacing`, and with additional attributes 'SLDF', 'spacingfactor' and possibly 'graph' (see linearmask).

Note

A valid shapefile includes at least three files with extensions '.shp', '.dbf' and '.shx'. The full name of the .shp file is provided as the 'file' argument of `read.linearmask`.

Line-level covariates are copied from SpatialLinesDataFrame input. Point-specific covariates may be added later using `covariates` or `addCovariates`.

See Also

`asgraph`, `linearmask`, `make.sldf`, `networkdistance`, `SpatialLinesDataFrame`

Examples

```
x <- seq(0, 4*pi, length = 200)
xy <- data.frame(x = x*100, y = sin(x)*300)
test <- read.linearmask(data = xy, spacing = 20)
plot(test)

xy2 <- data.frame(x = x*100, y = cos(x)*300)
test2 <- read.linearmask(data = xy2, spacing = 20)

plot(test2, add = TRUE)
```

---

showpath *Display Shortest Path*

Description

An interactive function to examine shortest paths and the associated network distances within a linear mask.

Usage

```
showpath(mask, add = FALSE, ...)
```

Arguments

`mask`	linearmask object
`add`	logical; if TRUE the plot is added to an existing plot
`...`	other arguments passed to `lines` for drawing network path

Details

The selected points are snapped to the nearest mask point before any calculations.

The path shown is a shortest path as determined by function `get.shortest.paths` of the **igraph** package. There may be others.

Value

Invisibly returns a list comprising

paths	a list of dataframes, each with the mask coordinates along a selected path
results	a dataframe with one row for each path giving the distances for each pair of points (columns 'from', 'to', 'Euclidean.d', 'network.d')

.

References

Csardi, G. and Nepusz, T. (2006) The igraph software package for complex network research, InterJournal, Complex Systems 1695. https://igraph.org/.

See Also

networkdistance

Examples

```
## Not run:

x <- seq(0, 4*pi, length = 200)
xy <- data.frame(x = x*100, y = sin(x)*300)
xy2 <- data.frame(x = x*100, y = cos(x)*300)
test <- read.linearmask(data = xy, spacing = 10)
test2 <- read.linearmask(data = xy2, spacing = 10)
test3 <- rbind(test, test2)
showpath(test3, lwd = 5)


## End(Not run)
```

Silverstream *Silverstream River Centrelines*

Description

Silverstream (45^{o}48' S, 170^{o}26' E) is a small river in a forested catchment near Dunedin, New Zealand, about 12 km from top to bottom. There are numerous side streams making this a good example of a dendritic network.

Format

An ESRI polyline shapefile (silverstream.shp, silverstream.shx, silverstream.dbf).

Source

Land Information New Zealand (LINZ) 1:50 000 topographic database https://data.linz.govt.nz/layer/50327-nz-river-centrelines-topo-150k/ licensed by LINZ for re-use under the Creative Commons Attribution 3.0 New Zealand licence. Downloaded 2014-10-26.

References

ESRI (1998) *ESRI shapefile technical description*. https://www.esri.com/content/dam/esrisites/sitecore-archive/Files/Pdfs/library/whitepapers/pdfs/shapefile.pdf.

Examples

```
## Not run:

inputdir <- system.file("extdata", package = "secrlinear")
silverstreammask <- read.linearmask(file = paste0(inputdir,"/silverstream.shp"),
    spacing = 50)
par(mar = c(1,1,1,1))
plot(silverstreammask)


## End(Not run)
```

---

sim.linearpopn *Simulate Animals on Lines*

Description

This function is a simple substitute for the **secr** function `sim.popn()` for the case of a linear habitat.

Usage

```
sim.linearpopn(mask, D, N, Ndist = c('poisson', 'fixed'), ...)
```

Arguments

`mask`	linearmask object
`D`	numeric density animals / km
`N`	number of individuals
`Ndist`	character string for distribution of total number of individuals
`...`	other arguments passed to `sim.popn`

Details

The linearmask input represents a discretized line - essentially a chain of line segments. By default, each segment is populated with a Poisson number of individuals. The user may specify `D` or `N`.

`D` may be a vector with one density per mask pixel, or a single number that will be applied across all pixels.

If `Ndist = 'fixed'` then a constant number of individuals N are simulated in each trial; otherwise N has a Poisson distribution across trials. N = sum(`D`) x mask length if `D` is specified.

This is a simplified wrapper for `sim.popn` called with `model2D = "linear"`.

Value

Object of class c('linearpopn', 'popn', 'data.frame').

Note

The population output from `sim.linearpopn` may be used unchanged with **secr** functions such as `sim.capthist`. However, to be faithful to the linear network you should set the 'userdist' argument of `sim.capthist` to `networkdistance`.

See Also

`linearpopn`, `sim.popn`

Examples

```
x <- seq(0, 4*pi, length = 200)
xy <- data.frame(x = x*100, y = sin(x)*300)
mask <- read.linearmask(data = xy, spacing = 10)
trps <- make.line(mask, n = 15, startbuffer = 1000, by = 30)

newmask <- clipmask(mask, trps, buffer = 200)

linpop <- sim.linearpopn(newmask, 200)
CH <- sim.capthist(trps, linpop, userdist = networkdistance)
plot(newmask)
plot(CH, add = TRUE)

secr.fit(CH, mask = mask, details = list(userdist = networkdistance))
```

---

subset.linearmask *Select Part of linearmask Object*

Description

Methods to subset linear mask objects.

Usage

```
## S3 method for class 'linearmask'
subset(x, subset, LineID, droplinesbeyond = Inf, ...)
```

Arguments

x	linearmask object from read.linearmask
subset	numeric or logical vector to select rows of mask
LineID	vector of identifiers for lines to retain
droplinesbeyond	logical; distance criterion for dropping lines (m)
...	other arguments (not used)

Details

Specify only one of 'subset' and 'LineID'.

If the input object has a 'graph' attribute then a new graph will be included in the output.

subset.linearmask leaves the underlying SpatialLinesDataFrame intact unless either (i) LineID is specified, or (ii) a finite value is provided for droplinesbeyond. droplinesbeyond retains lines that have at least one point within the specified distance of at least one mask point, after subsetting.

Value

A linearmask object.

See Also

read.linearmask, linearmask

Examples

```
## rbind two linear masks,
x <- seq(0, 4*pi, length = 200)
xy <- data.frame(x = x*100, y = sin(x)*300)
xy2 <- data.frame(x = x*100, y = cos(x)*300)
test <- read.linearmask(data = xy, spacing = 10)
test2 <- read.linearmask(data = xy2, spacing = 10)
test3 <- rbind(test, test2)
table(covariates(test3)$LineID)

## then retrieve one...
test4 <- subset(test3, LineID = '1.1')

## ... or the other
test5 <- subset(test3, LineID = '2.1')
```

`utility` *Utility Functions*

Description

Utilities for various functions

– conversion of a flat table of coordinates for a line or set of lines to a SpatialLinesDataFrame object

– snapping points from objects of various types (traps, popn, matrix etc.) to a linearmask object.

– converting distances along a linear mask to x-y coordinates

– build igraph network from linearmask object

Usage

```
make.sldf(coord, f)

snapPointsToLinearMask(xy, mask)

asgraph(mask)

getLineID(mask, laboffset = rep(spacing(mask) * 3, 2), ...)

branched(mask)
```

Arguments

`coord`	dataframe of coordinates - must include columns 'x','y'
`f`	vector of values by which to split coord rows
`xy`	any object that may be coerced to a 2-column matrix (columns interpreted as x- and y-coordinates)
`mask`	'linearmask' object
`laboffset`	offset of label from point
`...`	other arguments passed to `plot.linearmask`

Details

In `make.sldf`, if `f` is missing and there is a column 'lineID' in 'coord' then this will be used instead.

`snapPointsToLinearMask` shifts each point in the input to the nearest point on the graph stored as the 'graph' attribute of `mask`.

`make.line` creates a 'traps' object with equally spaced or clustered detectors.

`asgraph` is a utility function used internally for expressing the points in `mask` as an igraph network (Csardi and Nepusz 2006).

`getLineID` plots the mask and for each mouse click displays the corresponding point and LineID.

Other utilities specific to editing graphs are documented in edges.

Value

For make.sldf, a SpatialLinesDataFrame.

For snapPointsToLinearMask, the output has the same attributes (including class) as the xy input: only the coordinates themselves may change.

For alongmask, a 2-column dataframe with the x and y coordinates of the points.

For asgraph, an object of class 'igraph'

getLineID invisibly returns a dataframe with columns 'Point' and 'LineID'.

For branched, a logical value indicating whether the degree of any node exceeds 2.

References

Csardi, G. and Nepusz, T. (2006) The igraph software package for complex network research, InterJournal, Complex Systems 1695. https://igraph.org/.

See Also

checkmoves make.line, networkdistance, SpatialLinesDataFrame, showedges, deleteedges, addedges

Examples

```
## test make.sldf()

coord <- data.frame(x = c(1,2,3,1.05,2.05,3.05),
    y = c(3,2,2,3.05,2.05,2.05))

sldf1 <- make.sldf(coord)
sldf2 <- make.sldf(coord, f = c(1,1,1,2,2,2))

if (require('sp')) {
  plot(sldf1)
  plot(sldf2)
}

## test snapPointsToLinearMask()

## Not run:

x <- seq(0, 4*pi, length = 200)
xy <- data.frame(x = x*100, y = sin(x)*300)
mask <- read.linearmask(data = xy, spacing = 20)
plot(mask)
## click several points, right-click and select 'stop'
pts <- locator()
pts <- do.call(cbind, pts)
points(pts)
pts1 <- snapPointsToLinearMask(pts, mask)
segments(pts[,1], pts[,2], pts1[,1], pts1[,2])
points(pts1, pch = 16, col = 'red')
```

```
df <- data.frame(pts, pts1)
names(df) <- c('from.x', 'from.y', 'to.x', 'to.y')
df


## End(Not run)

## asgraph (called by networkdistance)
## see 'igraph' for further manipulation
grmask <- asgraph(glymemask)
if (require('igraph')) {
  summary(grmask)
}
```

Index