Package 'DSpat'

February 19, 2015

Type Package

Title Spatial Modelling for Distance Sampling Data

Version 0.1.6

Date 2014-12-08

Author Devin Johnson, Jeff Laake, Jay VerHoef

Maintainer Jeff Laake <Jeff.Laake@noaa.gov>

Description Fits inhomogeneous Poisson process spatial models
to line transect sampling data and provides estimate of
abundance within a region.

Depends R (>= 2.0.0), spatstat (>= 1.22-0), RandomFields

Imports rgeos, mgcv, sp

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2014-12-12 14:39:42

R topics documented:

DSpat-package   *Spatial modelling package for distance sampling data*

Description

DSpat uses the tools in spatstat to provide an analysis of distance sampling data in a spatial
context in which the density surface and the detection function are estimated simultaneously. The
package provides a fitted density surface and total abundance and measures of precision. It also
provides simulation capabilities.

Details

Package:	DSpat
Type:	Package
Version:	1.0
Date:	2008-04-08
License:	GPL version 2 or later

Conventional distance sampling (Buckland et al. 2001;2004) uses likelihood theory for estima-
tion of the detection function based on an assumed uniform distribution of perpendicular distances
within the transects. An adequate sampling design provides the basis for the uniform distribution
assumption and inference for abundance. No assumption is made about the spatial distribtion of the
object being sampled.

DSpat provides a full-likelihood framework for simultaneous estimation of the detection function
and abundance based on an inhomogeneous Poisson process. A full-likelihood approach has a
number of advantages because there is no strict requirement on the sampling design so it can be used
with unequal coverage sampling and it can provide estimates of the density surface and abundance
for any defined sub-area. Also, by modelling the observed data as a spatial process 'adjustments' to
the strip-width of the transect occur naturally when the transect extends beyond the area containing
objects. Consider sampling a marine environment with a contorted coastline such as fjords. In

sampling the open ocean, the full transect width can contain objects but within a fjord the strip is narrowed or clipped in areas where it extends onto land. This causes difficulties with conventional distance sampling which assumes a uniform distribution of objects across the entire strip. Thus, either a very narrow strip must be used for both areas or the detection function must be estimated separately for each region and even that can not completely cope with the problem. This variation in the spatial distribution of objects is handled easily with a spatial model that simultaneously estimates the detection function and the intensity (density) of the point process (e.g., animal/plant locations). The detection function is estimated as a covariate to explain the intensity of the observed point process as a function of perpendicular distance from the centerline. Thus, obviously the potential for confounding occurs if the pattern of transects is such that pattern of perpendicular distances is confounded with the spatial pattern of a covariate that determines the true intensity of the process. For example, if there was a density gradient with respect to the coastline and a single-sided transect parallel to the coastline was sampled then perpendicular distance and distance from the coastline are completely confounded. However, with a typical dual-sided transect, the pattern of perpendicular distance is no longer entirely confounded with the distance from the coastline because perpendicular increases away from the centerline in both directions. Thus, confounding would not occur except in the unlikely situation that intensity (density) varied relative to the coastline in such a fashion that was symmetric with respect to the centerline of the transect. With a modicum of care in the design, confounding between perpendicular distance and spatial covariates can be avoided but the analyst should always be cognizant of the potential for confounding.

Current Limitations: 1) assumes no overlap among strips 2) no handling of cluster size 3) assumes detection probability on the transect centerline is 1 4) can only use a detection function of the form $log(g(x)) = h(x)$ where $h(x)$ is linear in the parameters. For example, $h(x) = -tau * (distance^2)/2$. Note that any parameter such as tau is not constrained so this does allow for the possibility of an increasing detection function.

The first limitation will require some thought and work as we are unaware of any solutions at present. If there is overlap, when `owin` in `spatstat` is called with the `poly=transects`, the code will fail. It is easy to get around this problem to fit the model by using `study.area` as the boundary but the calls to `Kinhom` and `lgcp.correction` will not work properly. Also, there are some philosophical and inference issues that need to be considered if sample overlap. For example, is the point process fixed during sampling or should the replicate (and overlapping) samples be considered as independent realizations of the point process. Even though most designs do not have overlapping transects in theory, in practice if the line is composed of contiguous line segments that vary slightly in angle, the transects will overlap when created from the line segments. Some solution is needed as this is will likely occur in most real applications.

The latter three limitations can be resolved with the extension of the likelihood and additional coding in the package. DSpat currently uses `ppm` in `spatstat` which uses either `glm` or gam in `mgcv` to solve for the MLEs. We have functions that compute the likelihood and they can be generalized to accomodate these limitations but they have not been incorporated into the package yet.

DSpat relies heavily on the tools in `spatstat` and to a lesser degree the functions in `gpclib`, `mgcv` and `RandomFields`. DSpat provides aditional functions to cope with analysis and simulation of distance sampling data (line transect only at this stage). The functions in DSpat are listed below in categories with a brief description.

There are a number of concepts that should be understood prior to using this package. There are 2 coordinate systems that we will use. The first is the standard x,y Cartesian coordinate system with x on the horizontal and y on the vertical. The second which is not used extensively (yet) is the coordinate system within each line-transect. A line-transect is composed of a line (centerline)

which has a beginning (x0,y0) and end (x1,y1) and a rectangular strip with a defined width which extends width/2 to the left and right of the centerline. We use the term line to represent the line and transect for the rectangular strip (line-transect). The transect has a left-half and right-half defined by the direction of travel from beginning to the end of the line. Imagine the line-transect rotated such that it is vertical with the rotated versions of y0,y1 such that y0<y1 (travelling from south to north). We define a coordinate system u,v within the line-transect. The origin for u,v (u=0,v=0) is the rotated location of the beginning of the line (x0,y0) and u is equivalent to the standard horizontal x-coordinate with a range of (-width/2,width/2) and v is equivalent to the vertical y-coordinate with a range of (0,L) where L is the length of the line L=sqrt((x0-x1)^2+(y0-y1)^2). We use the variable distance for the perpendicular distance which is the absolute value of u.

So why have 2 coordinate systems? spatstat always works with the x,y coordinate system and it creates grids and the like with a horizontal-vertical orientation to the grid. In fitting distance sampling data we want to control the grid resolution relative to the u,v coordinates. In particular, we need to use a relatively fine grid in the u direction for estimation of the detection function which can change quickly over a small scale relative to most covariates that would be used for the intensity function. To use the spatstat code for grids and the like, we rotate the line-transects and observations to vertical from south to north and create the grid and counting weights in what is now the u,v coordinate system. Thus, for clarity we use a function argument epsvu in place of epsyx to show that the grid resolution is over u,v and not over x,y, unless the line-transects are all originally oriented vertically. Currently all line-transects must be rectangular we envision generalizing this and the u,v coordinate system will be used.

Even though all transects must be rectangular, the surveyed portion of the transect need not be rectangular. This is relevant when portions of the transect extend outside the boundary of the study area (defined region being sampled with the transects). The study area can be defined by any polygon as defined for class owin in spatstat. Note the restriction that the polygon coordinates must be given in a counter-clockwise direction. A simple example would be a square region such as

```
study.area=owin(xrange=c(0,100),yrange=c(0,100))
```

or a square with a missing portion

```
study.area=owin(poly=data.frame(x=c(0,40,40,100,100,0),                    y=c(0,0,50,50,
```

You can examine these by simply typing plot(study.area. Regardless, of the study area shape but depending on the orientation of the transects, portions of the transects can extend outside of the study area. For example, consider the corners of transects at a 45 degree angle extending across a rectangular study area. In many practical applications the width of the transect is so narrow relative to the dimensions of the study area, that these corners are of no consequence. However, in some applications with small scales this can be important. For example, surveys of narrow inlets (fjords) or rivers or contorted coastlines or surveys of small areas (see weeds).

This is handled in DSpat by clipping the portion of the rectangular transect that extends outside of the study area. The transects are clipped after they are rotated and gridded. This is important because that ensures the grids spatstat are positioned the same across all transects.

Analysis

Primary Functions:

dspat - main function for fitting spatial model to distance sampling data

integrate.intensity - computes predicted intensity surface, total abundance and precision with optional correction for over-dispersion

transect.intensity - computes predicted and observed counts within each transect in specified perpendicular distance intervals

Secondary Functions:

create.covariate.images - create a list of covariate images from a dataframe of covariates. The list of covariate images is used by LTDataFrame.

lines_to_strips - from a dataframe of lines this function creates a psp object and a list of transect polygons that assumes that lines are the centerlines of strips that have width as defined in the lines dataframe.

lgcp.correction - computes Monte Carlo correction for over-dispersion

LTDataFrame - assign covariates to the data (observations) and dummy points

quadscheme.lt - constructs a quadrature scheme for ppm that is more efficent for line transect samples which are small slices of the study area. The default quadrature scheme in spatstat creates dummy points across the entire study area which is terribly inefficient. This function rotates each line to vertical, creates a quadrature scheme within the line and then "rotates" back to original position to get the proper covariates. These line-by-line quadratures are then merged into a single quadrature.

Data preparation and utility

offset.points - this utility function is useful for most applied data sets in which the position of the observation is specified by the coordinates on the line that are perpendicular to the object. For a line and its observations, this function converts the object positions on the line and the perpendicular distance (negative is left) to the coordinates for the location of the object. It could be generalized to work with a radial distance and angle which would often be collected in shipboard work. It is also used from lines_to_strips to compute the vertices of the transect from the lines with a given width.

create.points.by.offset - this is a wrapper function that calls offset.points for each line in a lines dataframe and the corresponding observations in an observations dataframe, and returns a new observations dataframe with x,y being the true object coordinates.

dist2line - this function is the inverse of offset.points. It takes the true coordinates of points and a line and computes the perpendicular distance on the line and the coordinates on the line.

project2line - likewise this is a wrapper function for dist2line that is essentially the inverse of create.points.by.offset.

Internal

AIC.dspat - computes AIC for the model; only correct if a HPP or IPP process

coef.dspat -extracts the coefficients into a list with a vector for intensity coefficients and another for detection coefficients.

print.dspat - provides a listing of elements in the dspat object.

summary.dspat - extracts the ppm object and calls the spatstat summary function for this object.

vcov.dspat - extracts the variance-covariance matrix from the ppm object.

Ops.psp - allows syntax x==y or x!=y where x and y are psp objects.

rev_val - reorders vector for use in im.

im.clipped - fills in clipped image with vector of values defined over the clipped region.

owin.gpc.poly - converts first polygon in owin class to a gpc polygon.

Simulation

create.lines - create a systematic grid of parallel lines (with a random start) across a study area at a specified angle.

sample.points - extract observed points from a point process that fall within the defined set of strips and are randomly detected with a defined detection function.

simCovariates - a non-general function for simulating covariates in a 100x100 rectangle with discrete habitats and a linear vertical habitat feature. See DSpat.covariates.

simDSpat - a wrapper function to simulate distance sampling from a rectangular study area with a specified set of covariates on a grid. It calls create.lines, lines_to_strips, simPts and sample.points and returns a dataframe of lines and observations that can be used with the covariates datframe in dspat for an analysis.

simPts - creates a simulated point process in a study area by calling RFsimulate from the package RandomFields and rpoispp from spatstat. The intensity process is defined by a covariates dataframe and a formula and parameters for the intensity as a function of the covariates.

Example datasets

An example dataset from the fairy tale simulated world of simCovariates can be found in DSpat.obs, DSpat.lines, DSpat.covariates. To run an example analysis with these data, type example(dspat) or example(DSpat) to run the same code below.

An example real-world dataset of a devil's claw weed in a farm paddock can be found in weeds, weeds.all, weeds.obs, weeds.lines, weeds.covariates. To run a set of analyses, type example(weeds).

Author(s)

Devin S. Johnson, Jeffrey L. Laake, and Jay M. Ver Hoef

Maintainer: <Jeff.Laake@Noaa.Gov>

References

Johnson,D.S., Laake, J.L., and Ver Hoef, J.M. (in prep). A model based approach for making ecological inference from distance sampling data.

Buckland, S.T., D.R.Anderson, K.P. Burnham, J.L. Laake, D.L. Borchers, and L. Thomas. 2001. Introduction to Distance Sampling: Estimating Abundance of Biological Populations. Oxford University Press.

Buckland, S.T., D.R.Anderson, K.P. Burnham, J.L. Laake, D.L. Borchers, and L. Thomas. 2004. Advanced Distance Sampling. Oxford University Press.

See Also

spatstat

Examples

```
# get example data
data(DSpat.lines)
data(DSpat.obs)
```

```
data(DSpat.covariates)
# Fit model with covariates used to create the data
sim.dspat=dspat(~ river + factor(habitat),
                study.area=owin(xrange=c(0,100), yrange=c(0,100)),
                obs=DSpat.obs,lines=DSpat.lines,covariates=DSpat.covariates,
                epsvu=c(1,.01),width=0.4)
# Print
sim.dspat
# Summarize results
summary(sim.dspat)
# Extract coefficients
coef.intensity <- coef(sim.dspat)$intensity
coef.detection <- coef(sim.dspat)$detection
# Extract variance-covariance matrix (inverse information matrix)
J.inv <- vcov(sim.dspat)
# Compute AIC
AIC(sim.dspat)
# Visualize intensity (no. animals per area) and estimate abundance
mu.B <- integrate.intensity(sim.dspat,dimyx=100)
cat('Abundance =        ', round(mu.B$abundance,0), "\n")
dev.new()
plot(mu.B$lambda, col=gray(1-c(1:100)/120), main='Estimated Intensity')
plot(sim.dspat$model$Q$data,add=TRUE)
plot(owin(poly=sim.dspat$transect),add=TRUE)
plot(sim.dspat$lines.psp,lty=2,add=TRUE)
# Compute se and confidence interval for abundance without over-dispersion
mu.B <- integrate.intensity(sim.dspat,se=TRUE,dimyx=100)
cat("Standard Error =  ", round(mu.B$precision$se,0), "\n",
    "95 Percent Conf. Int. =   (", round(mu.B$precision$lcl.95,0), ',',
          round(mu.B$precision$ucl.95,0), ")", "\n")
```

---

create.covariate.images

Create a list of covariate images

Description

Creates a list of covariates images from a dataframe of covariates defined on a grid for the study
area. Images are created for variables contained in vector of names varnames and the values of the
covariates are in the covariates dataframe.

Usage

```
create.covariate.images(covariates, varnames)
```

Arguments

covariates	covariate dataframe (see DSpat for structure)
varnames	vector of names of fields contained in covariates that will be used for fitted model

Value

covariate.im - list of covariate images (class im)

Author(s)

Jeff Laake

---

create.lines   *Create a systematic sample of parallel lines across a grid*

Description

Create a systematic set of lines to sample a rectangular grid. The grid is positioned with a random start on the study area. The systematic grid can be set at any angle and the number of lines is set by the spacing or the spacing is set by width and number of lines. This is a wrapper function for `rlinegrid` in `spatstat`.

Usage

```
create.lines(study.area,nlines=NULL,width,spacing=NULL,angle=0)
```

Arguments

study.area	owin class defining area
nlines	number of lines
width	full transect width
spacing	spacing distance between centerlines
angle	angle of rotation in degrees anticlockwise from x-axis

Value

lines dataframe with label,x0,y0,x1,y1,width where x0,y0 is beginning and x1,y1 is end of the line

Author(s)

Jeff Laake

See Also

[simCovariates,simPts](#)

Examples

```
study.area=owin(xrange=c(0,100),yrange=c(0,100))
xp=create.lines(study.area,nlines=10,width=5,angle=180)
ls=lines_to_strips(xp,study.area)
plot(ls$lines,lty=2)
plot(owin(poly=ls$transects),add=TRUE)
```

---

create.points.by.offset

Create point dataframe offset from line

Description

For a set of observations with x,y locations on the line and a perpendicular distance, create a new observation dataframe with true x,y point locations.

Usage

```
create.points.by.offset(lines, observations)
```

Arguments

```
lines           - data frame of lines with the following structure
                          label - unique label
                          x0    - x coordinate of beginning of line
                          y0    - y coordinate of beginning of line
                          x1    - x coordinate of end of line
                          y1    - y coordinate of end of line
                          width - optional full width of each transect
                                     line is in center of transect
                          ...   - any number of covariates

observations    - data frame of observations with the following structure
                          label   - label linking it to a unique line
                          x       - x coordinate
                          y       - y coordinate
                          distance- perpendicular distance;
                                     positive=right side; negative=left side
                            ...   - any number of covariates
```

Value

observations dataframe with true x,y locations

Author(s)

Jeff Laake

See Also

offset.points

---

dist2line   *Compute perpendicular distances and projections onto line*

Description

Calculates perpendicular distances of a point process contained within a strip to the center line of the strip they are contained in. It also computes the positions of the objects projected onto the line. This is the inverse of the `offset.points` function.

Usage

```
dist2line(object.ppp, line.ends)
```

Arguments

object.ppp	point process for observations in a strip
line.ends	ends of line x0,y0,x1,y1

Value

distVals	vector of perpendicular distances
projection	dataframe of projected locations on the line

Author(s)

Devin Johnson

See Also

project2line

---

dspat   *Fits spatial model to distance sampling data*

Description

Creates a dspat object by fitting model represented by formula to observations along line transects in a study area with covariates defined for a grid over the entire study area.

Usage

```
dspat(int.formula=~1, det.formula=~1, study.area, obs, lines, covariates,
        epsvu=c(1,.01), width=NULL, use.gam=FALSE, show.warnings=FALSE,
        nclass=NULL)
```

Arguments

`int.formula`	formula for model of the point process intensity
`det.formula`	formula for interaction with distance in the detection process
`study.area`	owin class for study area
`obs`	dataframe of observations
`lines`	dataframe of lines
`covariates`	dataframe of covariates on a grid in the study area
`epsvu`	vector of height of pixels(y) and width of pixels(x)
`width`	full transect width; only needed if it is not specified in lines.df
`use.gam`	if TRUE uses gam instead of glm for fitting; if formula contains s() use.gam will be set TRUE by default
`show.warnings`	if TRUE, show the warnings created in building the quadrature.
`nclass`	number of distance classes for expected/observed counts.

Details

```
covariates has following structure
        x   - x coordinate of midpoint of grid cell
        y   - y coordinate of midpoint of grid cell
        ... - any number of covariate

  the data are ordered by column from left to right and
  from bottom to top such that y changes first from smallest
  to largest. Below are matrices showing y,x and their order
        3,1 3,2 3,3   3 6 9
        2,1 2,2 2,3   2 5 8
        1,1 1,2 1,3   1 4 7
```

The default for the intensity formula (`int.formula`) is ~1, a homogeneous Poisson process. Note that what is actually fitted is ~-1+constant where constant is 1 everywhere. This is done to avoid a glitch in `vcov.ppm`. The detection formula (`det.formula`) is expressed as a formula that interacts with $I(-distance^2/2)$. The default of ~1 is a detection function that is constant everywhere. If you use ~-1, it will drop distance which assumes a strip transect with perfect detection within the strip. The variables contained in `int.formula` must be all contained within `covariates` because they need to be defined across the entire study area. The variables contained in `det.formula` can be in `covariates` or in `lines` because for prediction of the intensity, distance is set to zero, so these covariates need not be known across the entire survey area.

The value of `epsvu[2]` is adjusted such that it is an even multiple of `width/2` so that the grid points are evenly distributed in the direction of perpendicular distance.

Value

list of class "dspat" with elements

`model`	output object from ppm
`lines.psp`	psp line segment process for center lines

transects	list of dataframes specifying polygonal transects
covariate.im	list of covariate images (class im)
study.area	owin class of study area
use.gam	TRUE if gam used and FALSE otherwise

Author(s)

Jeff Laake; Devin Johnson

See Also

[quadscheme.lt](),[LTDataFrame]()

Examples

```
# get example data
data(DSpat.lines)
data(DSpat.obs)
data(DSpat.covariates)
# Fit model with covariates used to create the data
sim.dspat=dspat(~ river + factor(habitat),
                study.area=owin(xrange=c(0,100), yrange=c(0,100)),
                obs=DSpat.obs,lines=DSpat.lines,covariates=DSpat.covariates,
                epsvu=c(4,.1),width=0.4)

# Print
sim.dspat
# Summarize results
summary(sim.dspat)
# Extract coefficients
coef.intensity <- coef(sim.dspat)$intensity
coef.detection <- coef(sim.dspat)$detection
# Extract variance-covariance matrix (inverse information matrix)
J.inv <- vcov(sim.dspat)
# Compute AIC
AIC(sim.dspat)
# Visualize intensity (no. animals per area) and estimate abundance
mu.B <- integrate.intensity(sim.dspat,dimyx=100)
cat('Abundance =        ', round(mu.B$abundance,0), "\n")
dev.new()
plot(mu.B$lambda, col=gray(1-c(1:100)/120), main='Estimated Intensity')
plot(sim.dspat$model$Q$data,add=TRUE)
plot(owin(poly=sim.dspat$transect),add=TRUE)
plot(sim.dspat$lines.psp,lty=2,add=TRUE)

# Compute se and confidence interval for abundance without over-dispersion
mu.B <- integrate.intensity(sim.dspat,se=TRUE,dimyx=100)
cat("Standard Error =  ", round(mu.B$precision$se,0), "\n",
    "95 Percent Conf. Int. =   (", round(mu.B$precision$lcl.95,0), ',',
          round(mu.B$precision$ucl.95,0), ")", "\n")
```

```
# Compute se and confidence interval for abundance with over-dispersion estimate
dev.new()
# The rest of the example has been put into a function to speed up package checking; remove # to run
# to run type do.dspat()
do.spat=function()
{
mu.B <- integrate.intensity(sim.dspat,se=TRUE,od=TRUE,reps=30,dimyx=100)
cat("Standard Error (corrected) =     ", round(mu.B$precision.od$se,0), "\n",
    "95 Percent Conf. Int. (corrected) =      (", round(mu.B$precision.od$lcl.95,0),
              ",", round(mu.B$precision.od$ucl.95,0), ")", "\n")
# Fit model with smooth of x and y
sim.dspat=dspat(~ s(x) + s(y),study.area=owin(xrange=c(0,100), yrange=c(0,100)),
                obs=DSpat.obs,lines=DSpat.lines,covariates=DSpat.covariates,
                epsvu=c(1,.01),width=0.4)
AIC(sim.dspat)
# Visualize intensity (no. animals per area) and estimate abundance
mu.B <- integrate.intensity(sim.dspat,dimyx=100,se=TRUE)
cat('Abundance =        ', round(mu.B$abundance,0), "\n")
cat("Standard Error =      ", round(mu.B$precision$se,0), "\n",
    "95 Percent Conf. Int. =      (", round(mu.B$precision$lcl.95,0),
              ",", round(mu.B$precision$ucl.95,0), ")", "\n")
dev.new()
plot(mu.B$lambda, col=gray(1-c(1:100)/120), main='Estimated Intensity')
plot(sim.dspat$model$Q$data,add=TRUE)
plot(owin(poly=sim.dspat$transect),add=TRUE)
plot(sim.dspat$lines.psp,lty=2,add=TRUE)
#
# Fit model with smooth of x and y with interaction
#
sim.dspat=dspat(~ s(x,y),study.area=owin(xrange=c(0,100), yrange=c(0,100)),
                obs=DSpat.obs,lines=DSpat.lines,covariates=DSpat.covariates,
                epsvu=c(1,.01),width=0.4)
AIC(sim.dspat)
# Visualize intensity (no. animals per area) and estimate abundance
mu.B <- integrate.intensity(sim.dspat,dimyx=100,se=TRUE)
cat('Abundance =        ', round(mu.B$abundance,0), "\n")
cat("Standard Error =      ", round(mu.B$precision$se,0), "\n",
    "95 Percent Conf. Int. =      (", round(mu.B$precision$lcl.95,0),
              ",", round(mu.B$precision$ucl.95,0), ")", "\n")
dev.new()
plot(mu.B$lambda, col=gray(1-c(1:100)/120), main='Estimated Intensity')
plot(sim.dspat$model$Q$data,add=TRUE)
plot(owin(poly=sim.dspat$transect),add=TRUE)
plot(sim.dspat$lines.psp,lty=2,add=TRUE)
}
```

DSpat.covariates   *Raster covariates study area*

Description

Example set of raster covariates for computing predicted intensity/abundance across the entire study area (100x100).

Usage

```
data(DSpat.covariates)
```

Format

A data frame with 10000 (1x1 raster element) on the following 4 variables.

x x coordinate for mid-point of raster cell

y y coordinate for mid-point of raster cell

river distance from river to center of raster element

habitat type of habitat for raster element

DSpat.lines   *Example DSpat lines dataframe*

Description

An example dataframe of 10 transect centerlines

Usage

```
data(DSpat.lines)
```

Format

A data frame with 10 observations on the following 5 variables.

label unique line label that links points to lines

x0 x-coordinate for beginning of line

x1 x-coordinate for end of line

y0 y-coordinate for beginning of line

y1 y-coordinate for end of line

Details

An example set of vertical lines evenly spaced across a window of 100x100.

DSpat.obs *Observation dataframe for DSpat*

Description

An example dataframe for the observations for fitting a spatial model with DSpat

Usage

```
data(DSpat.obs)
```

Format

A data frame with 395 observations on the following 6 variables.

label unique line label that links points to lines

x x coordinate of observation point

y y coordinate of observation point

Details

Example observation dataframe simulated with specific covariates across a 100x100 window. Only the x,y coordinates and line label are needed. The covariates are extracted based on the x,y coordinates.

---

integrate.intensity *Integrated intensity of fitted model*

Description

Compute intensity and its integration (abundance) and measures of precision with and without over-dispersion correction

Usage

```
integrate.intensity(x, dimyx=NULL, eps=NULL, se=FALSE, od=FALSE,
                    reps=100, silent=FALSE, J.inv=NULL, showplot=TRUE)
```

Arguments

x	dspat object
dimyx	number of y,x pixels
eps	height and width of pixels
se	if TRUE, compute std error of abundance and log-normal ci
od	if TRUE and se=TRUE, also compute over-dispersion corrected std error of abundance and log-normal ci
reps	number of reps for MC integration for over-dispersion correction
silent	if FALSE, show progress on MC integration
J.inv	var-cov matrix from fitted model
showplot	if TRUE show Poisson and empirical and fitted K-functions

Details

Either `dimyx` or `eps` can be specified. If neither specified then it uses the first covariate image in the dspat object to set the intensity grid. If more than one are specified then others are ignored with their priority for use matching the order they are listed above.

Value

Abundance	Estimate of expected abundance in the study area
distribution	dataframe containing N (predicted number of points in the cell),x,y (x,y coordinates of cell) and covariates used in the model
precision	List containing se, lcl.95, ucl.95, J.inv, and b.vec
precision.od	For over-dispersion estimate a list containing se, lcl.95, ucl.95, J.inv, and b.vec
lambda	estimated intensity image
W	window mask for study area

Author(s)

Devin Johnson; Jeff Laake

See Also

lgcp.correction

Internal   *Internal DSpat functions*

Description

Miscellaneous set of functions used in the package.

Usage

```
## S3 method for class 'dspat'
AIC(object,...,k)

## S3 method for class 'dspat'
print(x,...)

## S3 method for class 'dspat'
summary(object,...)

## S3 method for class 'dspat'
coef(object,...)

## S3 method for class 'dspat'
vcov(object,...)

x.psp==y.psp

x.psp!=y.psp

rev_val(x,y,val)

im.clipped(x, window)

owin.gpc.poly(window)
```

Arguments

x	for generic functions: a dspat object output from [dspat](#) with class dspat, for rev_val: a vector of x coordinates, for im.clipped: a vector of image values in order defined by spatstat
object	a dspat object output from [dspat](#) with class dspat
k	penalty per parameter in AIC, default is 2
x.psp, y.psp	psp objects
y	vector of y coordinates
val	vector of image values
window	class owin polygonal window
...	additional arguments for generic functions

Details

Internal functions:

AIC	Computes AIC value
print.dspat	print various objects in dspat object
summary.dspat	shows summary of ppm model object
coef.dspat	provides coefficients of the intensity and detection function
vcov.dspat	provides var-cov matrix of coefficients
Ops.psp	provides == and != operators for psp objects
rev_val	reverses order of `val` such that y increases within increasing x as needed in `im`
im.clipped	creates image and fills values (`val`) into the clipped portion of the image as defined by window
owin.gpc.poly	creates a gpc class poly from first polygon of an owin class

Author(s)

Jeff Laake

---

lgcp.correction	*Calculate Overdispersion factor for IPP fit via Monte Carlo Integration*

Description

Calculate Overdispersion factor for IPP fit via Monte Carlo Integration

Usage

```
lgcp.correction(fit.ppm, fit.lgcp, reps = 100, J.inv, silent = FALSE, lines.psp)
```

Arguments

fit.ppm	fitted model from ppm of spatstat
fit.lgcp	fitted model from lgcp.estK
reps	number of replicates for approximation
J.inv	variance-covariance matrix from fitted ppm model
silent	if FALSE, shows counter for replicates
lines.psp	line segment process

Value

J.inv.corr	Adjusted var-cov matrix
u	score matrix

Author(s)

Devin Johnson

See Also

integrate.intensity

---

lines_to_strips   *Convert lines to transects (strips)*

Description

Convert lines (center) with transect widths to strips and compute rotation angle from vertical. With the intersect function in gpclib, it also now clips the portions of the transects that are outside the study area.

Usage

```
lines_to_strips(lines, study.area, width = NULL)
```

Arguments

lines	dataframe with fields named label,x0,x1,y0,y1 and optionally width
study.area	owin class giving study area window
width	optional; if all lines have the same width it can be specified here

Details

The function assumes that the intersection of the strip and the study area only results in a single intersection polygon. That means the entire strip cannot pass outside the study area and then come back into the study area as in an aerial transect that passes over water to over land and then back over water. In this case, the line should end when it passes out of the sampled area and restarted when back in the sampled area.

Value

lines	a psp class of lines with label and angles for rotation added
transects	a list of dataframes with polygon coordinates

Author(s)

Jeff Laake

LTDataFrame   *Creates covariate dataframes*

Description

Creates covariate dataframes for observations and dummy quadrature points

Usage

```
LTDataFrame(study.area, lines, lines.psp, int.formula, det.formula,
                covariates, Q.lt)
```

Arguments

study.area	owin object that defines study area
lines	data frame of lines with structure as shown in `quadscheme.lt`
lines.psp	psp class with added list elements `width` and `label`
int.formula	model formula for intensity process
det.formula	model formula for detection scale process
covariates	covariate dataframe (see DSpat for structure)
Q.lt	lt quadscheme of class quad

Details

Checks to make sure that all of the variables used in formula are either in `covariates` or in `lines`. Then it extracts the values of `covariates` for each observation and for dummy points. These are merged with the needed covariates from `lines` and then a single dataframe is returned with the observations followed by the dummy points. In addition, the covariate images in a list are returned to keep with the `dspat` object for use in `integrate.intensity`.

Value

cov.df	dataframe of covariates followed by rows for covariates for dummy quadrature points
covariate.im	list of covariate images

Author(s)

Devin Johnson; Jeff Laake

See Also

`quadscheme.lt`,`create.covariate.images`

---

offset.points *Offset points from the line to actual position*

Description

Convert x,y point locations on the line and a distance (negative is left of line for the direction of travel) for a series of points in a strip.

Usage

```
offset.points(line, pts)
```

Arguments

line   vector with named components of x0,y0,x1,y1; line traverses from (x0,y0) to (x1,y1)

pts   dataframe of x,y,distance for each observed point; x,y is the location on the line that is perpendicular to the object; a negative distance implies it is on the left side of the line as defined by the direction of travel

Value

pts dataframe with x,y locations of the objects offset from the line at the appropriate distance and side.

Author(s)

Jeff Laake

See Also

create.points.by.offset

---

project2line *Project points onto line*

Description

Projects point process contained in strips to the center line of each strip containing points. This is the inverse of the create.points.by.offset function.

Usage

```
project2line(obs.ppp, lines.psp)
```

Arguments

`obs.ppp`	point process contained in strips
`lines.psp`	line segment process with label field

Value

dataframe of projected locations (x,y) on the lines

Author(s)

Jeff Laake

See Also

dist2line

---

quadscheme.lt	*Create line transect quadrature for spatstat*

Description

Creates a quadrature for spatstat from a study area, observations, and lines

Usage

```
quadscheme.lt(study.area, observations, lines, width = NULL,
                 epsvu = c(1, 0.01), show.warnings=FALSE)
```

Arguments

`study.area`	owin class giving the boundaries of the study area
`observations`	- data frame of observations with the following structure

```
                        label  - label linking it to a unique line
                        x       - x coordinate
                        y       - y coordinate
                        distance- perpendicular distance from center line
                        ...     - any number of covariates
```

`lines`	- data frame of lines with the following structure

```
                        label - unique label
                        x0    - x coordinate of beginning of line
                        y0    - y coordinate of beginning of line
                        x1    - x coordinate of end of line
                        y1    - y coordinate of end of line
                        width - optional full width of transect around line
                        angle - angle of rotation to get to vertical
                        ...   - any number of covariates
```

width	if no width field is given in lines then it must be specified here as a constant width for all lines
epsvu	pixel dimensions epsvu[1] in v direction (height) and epsvu[2] in u direction (width) (these are used once line is rotated vertically.
show.warnings	if TRUE, warnings from quadrature construction will be shown.

Value

Q	quadscheme as defined in spatstat
transects	list of transect polygon dataframes
lines.psp	line segment process

Author(s)

Devin Johnson; Jeff Laake

See Also

LTDataFrame

---

sample.points	*Sample points within each transect and filter with specified detection function*

Description

Create a dataframe of observations by simulating distance sampling of a point process with a systematic set of lines over a rectangular grid. The transects, lines and point process(points.ppp) are input arguments. Detection of observations is specified with a user-defined detection function which takes a distance vector and set of parameters det.par as its arguments.

Usage

```
sample.points(transects,lines,points.ppp,detfct=NULL,det.par=NULL,
                    det.formula=~1,covariates=NULL)
hndetfct(x,scale)
```

Arguments

transects	list of transect polygons
lines	dataframe of lines
points.ppp	simulated point process
detfct	detection function name
det.par	parameters for the detection function

det.formula	formula of covariates to use for scale of distance if det.formula=~-1, uses a strip transect
covariates	a matrix with columns x,y and any number of covariates x and y are the mid points of the grid cells; the order of the rows must match the formulation for function im
x	perpendicular distance for detection function
scale	scale for detection function

Details

Definition for half-normal detection function (hndetfct) is exp(-(x^2/(2*exp(scale)^2)))

Value

observation dataframe with fields label,x,y,distance for line label, x,y coordinates of the observation location and its perpendicular distance from the line

Author(s)

Jeff Laake

See Also

[simCovariates,simPts,create.lines](#)

Examples

```
study.area=owin(xrange=c(0,100),yrange=c(0,100))
hab.range=30
probs=c(1/3,2/3)
covariates = simCovariates(hab.range, probs)
xlines=create.lines(study.area,nlines=10,width=5,angle=45)
ls=lines_to_strips(xlines,study.area)
plot(ls$lines,lty=2)
plot(owin(poly=ls$transects),add=TRUE)
xpp=simPts(covariates=covariates,int.formula=~factor(habitat),int.par=c(0,1,2),EN=1000)
obs=sample.points(transects=ls$transects,lines=xlines,points.ppp=xpp,
                  hndetfct,c(1),covariates=covariates)
plot(ppp(x=obs$x,y=obs$y,window=study.area),add=TRUE,pch=20)
```

---

simCovariates   *Simulates covariates for an example in DSpat*

Description

Create a set of covariates in a 100x100 world with a vertical linear feature and discrete habitats.

Usage

```
simCovariates(hab.range=30, probs=c(1/3,2/3), river.loc=50)
```

Arguments

`hab.range`	habitat range that controls patchiness
`probs`	ordered probablities that define habitat cutoffs
`river.loc`	x coordinate for north-south river location

Details

The number of habitat types is the length of `probs` plus 1. The habitats are stored as a numeric from 1 to the number of types, but should be fitted with habitat as a factor variable. The distance to the river is a scaled distance from 0 to 1.

Value

dataframe with columns x,y,river and habitat

Author(s)

Devin Johnson; Jeff Laake

See Also

[simPts](#)

Examples

```
covariates = simCovariates(hab.range=50, probs=c(1/3,2/3,7/8))
```

---

simDSpat	*Simulate a distance sample from a specified spatial point process*

Description

This is a wrapper function that calls all of the functions needed to simulate and sample a point process over a defined `study.area` with a specified `covariates` on a grid. In sequence it calls `create.lines`, `lines_to_strips`, `simPts`, and `sample.points`.

Usage

```
simDSpat(study.area=owin(xrange=c(0,100),yrange=c(0,100)),covariates,
         angle=90,nlines=10,spacing=10,width=1,int.formula=~1,
         int.par=1,model="exp",cor.par=NULL,EN=1000,detfct=hndetfct,
         det.formula=~1,det.par=log(width/3),showplot=FALSE,showlines=FALSE,
         showpts=FALSE,pts=NULL,...)
```

Arguments

study.area	owin class defining area
covariates	a matrix with columns x,y and any number of covariates x and y are the mid points of the grid cells; the order of the rows must match the formulation for function im
angle	angle of rotation in degrees anticlockwise from x-axis
nlines	number of lines
spacing	spacing distance between centerlines
width	full transect width
int.formula	formula for deriving expected intensity from covariates
int.par	parameters for intensity formula
model	either "exp" or "gauss" for exponential or Gaussian correlation
cor.par	parameters controlling clustering of points cor.par[1] $sigma^2$ cor.par[2]=$alpha$ where cov(y1,y2)=$sigma^2 * exp(-d^p/alpha)$ and d is the distance between y1 and y2 and p=1 for exp and p=2 for gauss; if it is not specified then no additional clustering is included.
EN	expected number of points
detfct	detection function name
det.formula	formula of covariates to use for scale of distance if det.formula=~-1, uses a strip transect
det.par	parameters for the detection function
showplot	if TRUE show plot of the simulated points
showlines	if TRUE show lines and transects on the plot
showpts	if TRUE show points on the plot
pts	if not NULL use these points rather than generating new ones; this allows generation of a single set of points and evaluation of different sampling designs or intensity
...	parameters, if any, passed to plot

Value

a list with elements

lines	lines dataframe with label,x0,y0,x1,y1,width where x0,y0 is beginning and x1,y1 is end of the line
observations	a dataframe of the coordinates of the observed points

Author(s)

Jeff Laake

See Also

simCovariates,simPts

Examples

```
# Code stored in a function to speed up package checking run by typing do.simDSpat
# Some portions of this code will not pass the packge check on Linux and Mac and will issue
# an error that the polygons intersect even though when run as an example, the
# error is not encountered; so polygon checking is turned off
do.simDSpat=function()
{
# Now that it is in a function shouldn't need following line
spatstat.options(checkpolygons=FALSE)
study.area=owin(poly=list(x=c(0,40,40,100,100,0),y=c(0,0,40,40,100,100)))
covariates = simCovariates(hab.range=30, probs=c(1/3,2/3))
simdata=simDSpat(study.area,covariates,int.formula=~factor(habitat),
                 int.par=c(0,1,2),angle=45,nlines=10,width=3,det.par=.1)
sim.dspat=dspat(int.formula=~factor(habitat),study.area=study.area,
            obs=simdata$observations,lines=simdata$lines,
            covariates=covariates,epsvu=c(1,.05))
summary(sim.dspat)
AIC(sim.dspat)
coef(sim.dspat)
mu.B <- integrate.intensity(sim.dspat,dimyx=100,se=TRUE)
cat('Abundance =       ', round(mu.B$abundance,0), "\n")
cat('Standard Error =  ', round(mu.B$precision$se,0), "\n",
    '95 Percent Conf. Int. =  (', round(mu.B$precision$lcl.95,0), ',',
         round(mu.B$precision$ucl.95,0), ')', '\n')

mu.B <- integrate.intensity(sim.dspat,dimyx=100,se=TRUE,od=TRUE,reps=50)
cat('Abundance =       ', round(mu.B$abundance,0), "\n")
cat('Standard Error (corrected) =  ', round(mu.B$precision.od$se,0), "\n",
    '95 Percent Conf. Int.(corrected)  =  (', round(mu.B$precision.od$lcl.95,0), ',',
         round(mu.B$precision.od$ucl.95,0), ')', '\n')
plot(mu.B$lambda, main='Estimated Intensity')
plot(sim.dspat$lines.psp,lty=2,add=TRUE)
plot(owin(poly=sim.dspat$transect),add=TRUE)
plot(sim.dspat$model$Q$data,add=TRUE)
# Now sample with same point process realization with a different sampling angle
dev.new()
simdata=simDSpat(study.area,covariates,int.formula=~factor(habitat),
                 int.par=c(0,1,2),angle=90,nlines=10,width=3,pts=simdata$pts)
sim.dspat=dspat(int.formula=~factor(habitat),study.area=study.area,
            obs=simdata$observations,lines=simdata$lines,
            covariates=covariates,epsvu=c(1,.05))
mu.B <- integrate.intensity(sim.dspat,dimyx=100,se=TRUE)
cat('Abundance =       ', round(mu.B$abundance,0), "\n")
cat('Standard Error =  ', round(mu.B$precision$se,0), "\n",
    '95 Percent Conf. Int. =  (', round(mu.B$precision$lcl.95,0), ',',
         round(mu.B$precision$ucl.95,0), ')', '\n')
mu.B <- integrate.intensity(sim.dspat,dimyx=100,se=TRUE,od=TRUE,reps=50)
cat('Abundance  =       ', round(mu.B$abundance,0), "\n")
cat('Standard Error (corrected)=  ', round(mu.B$precision.od$se,0), "\n",
    '95 Percent Conf. Int. (corrected)=  (', round(mu.B$precision.od$lcl.95,0), ',',
         round(mu.B$precision.od$ucl.95,0), ')', '\n')
plot(mu.B$lambda, main='Estimated Intensity')
```

```
plot(sim.dspat$lines.psp,lty=2,add=TRUE)
plot(owin(poly=sim.dspat$transect),add=TRUE)
spatstat.options(checkpolygons=TRUE)
plot(sim.dspat$model$Q$data,add=TRUE)
# Sample with detection as a function of habitat
dev.new()
study.area=owin(poly=list(x=c(0,40,40,100,100,0),y=c(0,0,40,40,100,100)))
simdata=simDSpat(study.area,covariates,int.formula=~factor(habitat),
                 int.par=c(0,1,2),angle=45,nlines=10,width=3,
                 det.par=c(.1,.5,-.2),det.formula=~factor(habitat))
sim.dspat=dspat(int.formula=~factor(habitat),det.formula=~factor(habitat),
                study.area=study.area, obs=simdata$observations,lines=simdata$lines,
                covariates=covariates,epsvu=c(1,.05))
summary(sim.dspat)
AIC(sim.dspat)
coef(sim.dspat)
mu.B <- integrate.intensity(sim.dspat,dimyx=100,se=TRUE)
cat('Abundance =       ', round(mu.B$abundance,0), "\n")
cat('Standard Error =  ', round(mu.B$precision$se,0), "\n",
    '95 Percent Conf. Int. =   (', round(mu.B$precision$lcl.95,0), ',',
          round(mu.B$precision$ucl.95,0), ')', '\n')
plot(mu.B$lambda, main='Estimated Intensity')
plot(sim.dspat$lines.psp,lty=2,add=TRUE)
plot(owin(poly=sim.dspat$transect),add=TRUE)
plot(sim.dspat$model$Q$data,add=TRUE)
###########################################################
# Generate example like Figure used in paper for simulations
# Note: it required a patch to plot.im from spatstat to
# fix the ribbon bar on the side.
#
#
#              if(is.null(list(...)$zlim))
#            {
#                ribbonvalues <- seq(vrange[1], vrange[2], length = ribn)
#                ribbonrange <- vrange
#                ribbonticks <- clamp(pretty(ribbonvalues), vrange)
#            }
#            else
#            {
#                zlim=list(...)$zlim
#                ribbonvalues <- seq(zlim[1], zlim[2], length = ribn)
#                ribbonrange <- zlim
#                ribbonticks <- clamp(pretty(ribbonvalues), zlim)
#            }
#
############################################################
study.area=owin(poly=list(x=c(0,100,100,0),y=c(0,0,100,100)))
covariates = simCovariates(hab.range=30, probs=c(1/3,2/3))
postscript("Figure1.eps",horizontal=FALSE)
par(mfrow=c(2,1),mar=c(3, 1, 3, 1) + 0.1)
k=10
width=0.04*100/k
En=75
```

```
p=0.25
EN=En/(.04*p)
simdata=simDSpat(study.area,covariates,int.formula=~factor(habitat)+river,EN=EN,
                 int.par=c(0,1,2,-1),angle=90,nlines=k,width=width,
                 det.par=log(width/5),showplot=TRUE,col=gray(1-c(1:100)/120),
                 breaks=(0:100)*2.5/100,
                 zlim=c(0,2.5))

lines(c(50,50),c(0,100),lty=2)
sim.dspat=dspat(int.formula=~factor(habitat)+river,study.area=study.area,
            obs=simdata$observations,lines=simdata$lines,
            covariates=covariates,epsvu=c(1,width/100))
summary(sim.dspat)
AIC(sim.dspat)
coef(sim.dspat)
mu.B <- integrate.intensity(sim.dspat,dimyx=100)
plot(mu.B$lambda, col=gray(1-c(1:100)/120), main='Estimated Intensity',
  breaks=(0:100)*2.5/100,zlim=c(0,2.5))
plot(sim.dspat$lines.psp,lty=2,add=TRUE)
plot(owin(poly=sim.dspat$transect),add=TRUE)
plot(sim.dspat$model$Q$data,add=TRUE)
dev.off()
spatstat.options(checkpolygons=TRUE)
}
```

---

simPts	*Simulates point process on a rectangular grid*

Description

Generates a set of points from either a homogeneous or inhomogeneous Poisson process with optional clustering. This is a wrapper function for rpoispp from the spatstat package. The intensity is defined by covariates on a grid, an intensity formula and parameters. The correlation structure is defined by the model and the correlation parameters.

Usage

```
simPts(covariates,int.formula=~1, int.par=c(1), EN=100,
             model, showplot=FALSE, showpts=FALSE, ...)
```

Arguments

covariates	a matrix with columns x,y and any number of covariates x and y are the mid points of the grid cells; the order of the rows must match the formulation for function im
int.formula	formula for deriving expected intensity from covariates
int.par	parameters for intensity formula
EN	expected number of points

model	a spatial model for suimulation. See [RFsimulate](#) for further specification.
showplot	if TRUE, plot intensity and point process
showpts	if TRUE show points on the plot
...	parameters, if any, passed to plot

Value

ppp object of point locations

Author(s)

Devin Johnson; Jeff Laake

See Also

[simCovariates](#)

Examples

```
hab.range=30
probs=c(1/3,2/3)
covariates = simCovariates(hab.range, probs)
xpp=simPts(covariates=covariates,int.formula=~factor(habitat),int.par=c(0,1,2))
plot(xpp)
```

---

transect.intensity   *Compute expected and observed counts by distance within transect*

Description

Computes the expected and observed counts for equally-spaced bins of perpendincular distance within each transect. Expected and observed counts are each a matrix with a row for each transect and a column for each distance bin.

Usage

```
transect.intensity(x, epsvu=NULL, obs.ppp, covariates, nclass=NULL, width)
```

Arguments

x	dspat object
epsvu	epsvu setting for fitted model; only uses epsvu[2] value for u
obs.ppp	observation point process
covariates	dataframe of covariates at quadrature points
nclass	number of equally-spaced distance intervals within 0-width/2
width	maximum full transect width over all transects

Details

The actual number of distance bins will only match nclass if it is selected such that nclass*epsvu[2] is an even multiple of width/2. The function [dspat](#) adjusts epsvu[2] such that it is an even multiple of width/2 and this function assumes that condition holds. Sometimes your choice of epsvu[2] will provide less than optimal choices for nclass and in some cases it can only choose a single bin. In these cases, select another value of epsvu[2] which is a multiple for width/2.

If nclass is not specified then it uses the default of ceiling(sqrt(n)) intervals.

Value

exp.counts	matrix of expected counts in each distance bin (columns) for each transect (row)
obs.counts	matrix of observed counts in each distance bin (columns) for each transect (row)

Author(s)

Jeff Laake

---

weeds	*Dubbo weed data*

Description

Locations of devils claw in a farming paddock. Locations to all weeds are given and those observed along one of eight 150m wide transects (75m each side) are specified as Seen=1.

Usage

```
data(weeds)
```

Format

A data frame with 742 observations on the following 4 variables.

Transect Label of the transect 1 to 8

SignedDistance perpendicular distance in meters of weed from centerline; negative left and positive right

Distance absolute perpendicular distance

Seen weed was seen if 1 and 0 if missed

Details

These are the data that were provided by Melville and Welsh (see reference below) that were used in their Biometrics paper on distance sampling. In their paper they specified that the transects were laid out parallel in a north-south direction and presumably the transects were contiguous. This allows us to construct an x coordinate for each weed but no y coordinate was provided. In our use of these data we have created a y coordinate using runif and we have assumed the entire study area was 1200x1200 or 1.44 sq kilometers. They also stated that on transect 5-8 sheep ate the leafy part of the weed but there was no sheep grazing on transects 1-4. Presumably there was a fence between the sets of transects.

References

Melville, G. J., and A. H. Welsh. 2001. Line transect sampling in small regions. Biometrics 57:1130-1137.

Examples

```
###########################################################################
# Dubbo weed data
###########################################################################
#
# Example creates a function that you can run.  It is not run as
# part of the exampled to speed up package checking
# To run, code type do.weeds()
do.weeds=function()
{
  data(weeds.all)
  TrueAbundance=dim(weeds.all)[1]
  cat("\nTrue N= ",TrueAbundance,"\n")
  study.area=owin(xrange=c(0,1200),yrange=c(0,1200))
  data(weeds.lines)
  data(weeds.obs)
  data(weeds.covariates)
  study.area=owin(xrange=c(0,1200),yrange=c(0,1200))
#
# The entire study area is covered by the 8 N-S strips that are each 150m wide
# Sheep are absent on strips 1-4 and present on strips 5-8
# The following fits a model using all weeds whether they were seen or not
#
  weeds.dspat=dspat(int.formula=~factor(strip),det.formula=~-1,
                    study.area=study.area,
                    obs=weeds.all,lines=weeds.lines,covariates=weeds.covariates,
                    epsvu=c(100,1))
  mu.B <- integrate.intensity(weeds.dspat,dimyx=120,se=TRUE)
  cat('Abundance =        ', round(mu.B$abundance,0), "\n")
  pdf("TrueIntensity.pdf")
  plot(mu.B$lambda, main='True intensity by strip')
  plot(weeds.dspat$lines.psp,lty=2,add=TRUE)
  plot(owin(poly=weeds.dspat$transect),add=TRUE)
  plot(weeds.dspat$model$Q$data,add=TRUE,pch=20)
  dev.off()
# Compute distances for each weed
  obs.ppp=weeds.dspat$model$Q$data
  no.sheep.distances=NULL
  sheep.distances=NULL
  transects=weeds.dspat$transects
  for (i in 1:4)
    no.sheep.distances=c(no.sheep.distances,
     dist2line(obs.ppp[owin(poly=transects[i])],weeds.dspat$lines.psp$ends[i,])$distance)
  sheep.distances=NULL
  for (i in 5:8)
    sheep.distances=c(sheep.distances,
```

```
       dist2line(obs.ppp[owin(poly=transects[i])],weeds.dspat$lines.psp$ends[i,])$distance)
    pdf("True Distance Distribution.pdf")
    par(mfrow=c(2,1))
   hist(no.sheep.distances,breaks=(0:15)*5,main="Sheep absent",xlab="Perpendicular distance (m)")
   hist(sheep.distances,breaks=(0:15)*5,main="Sheep present",xlab="Perpendicular distance (m)")
    dev.off()
    no.sheep=hist(no.sheep.distances,breaks=(0:15)*5,plot=FALSE)$counts
    with.sheep=hist(sheep.distances,breaks=(0:15)*5,plot=FALSE)$counts
 # summary of abundance per strip
    Est.N=by(mu.B$distribution$N,cut(mu.B$distribution$x,seq(0,1200,150)),sum)
    True.N=by(weeds.all$x,cut(weeds.all$x,seq(0,1200,150)),length)
    pdf("TrueAbundanceByStrip.pdf")
    barplot(rbind(True.N,Est.N),beside=TRUE,legend=TRUE,names.arg=1:8,main="All weeds")
    dev.off()
 #  The following code will produce the true detection probability as a function of
 #  distance for no sheep (lines 1-4) and sheep (lines 5-8) using all known weed locations
 #  observed weed locations.
    sheep.labels.obs=cut(weeds.obs$label,c(1,4,8),include.lowest=TRUE)
    levels(sheep.labels.obs)=c("Sheep absent","Sheep present")
    sheep.labels=cut(weeds.all$label,c(1,4,8),include.lowest=TRUE)
    levels(sheep.labels)=c("Sheep absent","Sheep present")
    cat("\n All weeds \n")
    table(sheep.labels,cut(weeds.all$distance,(0:10)*7.5,include.lowest=TRUE))
    det=table(sheep.labels.obs,cut(weeds.obs$distance,(0:10)*7.5,include.lowest=TRUE))/
    table(sheep.labels,cut(weeds.all$distance,(0:10)*7.5,include.lowest=TRUE))
    cat("\n Detection \n")
    det
    pdf("TrueDetection.pdf")
    barplot(det,beside=TRUE,main="Dubbo weed detection probability",
              xlab="Perpendicular distance",legend=TRUE)
    dev.off()
 #
 # For the observed weeds with N-S transects:
 #
 # 6 different models were fit for each pairing of:
 #  int.formula:
 #  3 formulas for intensity: ~factor(sheep), ~factor(strip), ~s(x)
 #  det.formula
 #  2 formulas for detection: ~1 (constant sigma), ~factor(sheep) (sigma for sheep,no sheep)
 #
 # A half-normal detection function is assumed which is fitted with I(-distance^2/2)
 #
 # Fit model ~sheep, ~1
 weeds.dspat.1=dspat(int.formula=~factor(sheep), study.area=study.area,
                     obs=weeds.obs,lines=weeds.lines,covariates=weeds.covariates,
                     epsvu=c(100,1))
 AIC(weeds.dspat.1)
 coef(weeds.dspat.1)
 mu.B = integrate.intensity(weeds.dspat.1,dimyx=120,se=TRUE)
 cat('Abundance =      ', round(mu.B$abundance,0), "\n")
 cat('Standard Error =  ', round(mu.B$precision$se,0), "\n",
     '95 Percent Conf. Int. =  (', round(mu.B$precision$lcl.95,0), ',',
           round(mu.B$precision$ucl.95,0), ')', '\n')
```

```
pdf("NS_model_1_intensity.pdf")
plot(mu.B$lambda, main='Estimated Intensity')
plot(weeds.dspat.1$lines.psp,lty=2,add=TRUE)
plot(owin(poly=weeds.dspat.1$transect),add=TRUE)
plot(weeds.dspat.1$model$Q$data,add=TRUE,pch=20)
dev.off()
# Fit model ~sheep, ~sheep
weeds.dspat.2=dspat(int.formula=~factor(sheep),det.formula=~factor(sheep),
                    study.area=study.area,
                    obs=weeds.obs,lines=weeds.lines,covariates=weeds.covariates,
                    epsvu=c(100,1))
summary(weeds.dspat.2)
AIC(weeds.dspat.2)
coef(weeds.dspat.2)
mu.B = integrate.intensity(weeds.dspat.2,dimyx=120,se=TRUE)
cat('Abundance =        ', round(mu.B$abundance,0), "\n")
cat('Standard Error =  ', round(mu.B$precision$se,0), "\n",
    '95 Percent Conf. Int. =   (', round(mu.B$precision$lcl.95,0), ',',
          round(mu.B$precision$ucl.95,0), ')', '\n')
pdf("NS_model_2_intensity.pdf")
plot(mu.B$lambda, main='Estimated Intensity')
plot(weeds.dspat.2$lines.psp,lty=2,add=TRUE)
plot(owin(poly=weeds.dspat.2$transect),add=TRUE)
plot(weeds.dspat.2$model$Q$data,add=TRUE,pch=20)
dev.off()
# Fit model ~factor(strip), ~1
weeds.dspat.3=dspat(~factor(strip),study.area=study.area,
                    obs=weeds.obs,lines=weeds.lines,covariates=weeds.covariates,
                     epsvu=c(100,1))
summary(weeds.dspat.3)
AIC(weeds.dspat.3)
coef(weeds.dspat.3)
mu.B = integrate.intensity(weeds.dspat.3,dimyx=120,se=TRUE)
cat('Abundance =        ', round(mu.B$abundance,0), "\n")
cat('Standard Error =  ', round(mu.B$precision$se,0), "\n",
    '95 Percent Conf. Int. =   (', round(mu.B$precision$lcl.95,0), ',',
          round(mu.B$precision$ucl.95,0), ')', '\n')
pdf("NS_model_3_intensity.pdf")
plot(mu.B$lambda, main='Estimated Intensity')
plot(weeds.dspat.3$lines.psp,lty=2,add=TRUE)
plot(owin(poly=weeds.dspat.3$transect),add=TRUE)
plot(weeds.dspat.3$model$Q$data,add=TRUE,pch=20)
dev.off()
# Fit model ~factor(strip), ~factor(sheep)
weeds.dspat.4=dspat(int.formula=~factor(strip),det.formula=~factor(sheep),
                    study.area=study.area,
                    obs=weeds.obs,lines=weeds.lines,covariates=weeds.covariates,
                    epsvu=c(100,0.75),nclass=10)
summary(weeds.dspat.4)
AIC(weeds.dspat.4)
coef(weeds.dspat.4)
mu.B = integrate.intensity(weeds.dspat.4,dimyx=120,se=TRUE)
mu.B.4=mu.B
```

```
cat('Abundance =        ', round(mu.B$abundance,0), "\n")
cat('Standard Error =  ', round(mu.B$precision$se,0), "\n",
    '95 Percent Conf. Int. =  (', round(mu.B$precision$lcl.95,0), ',',
            round(mu.B$precision$ucl.95,0), ')', '\n')
pdf("NS_model_4_intensity.pdf")
plot(mu.B$lambda, main='Estimated Intensity')
plot(weeds.dspat.4$lines.psp,lty=2,add=TRUE)
plot(owin(poly=weeds.dspat.4$transect),add=TRUE)
plot(weeds.dspat.4$model$Q$data,add=TRUE,pch=20)
dev.off()
# Fit model ~s(x), ~1
weeds.dspat.5=dspat(int.formula=~s(x),
                    study.area=study.area,
                    obs=weeds.obs,lines=weeds.lines,covariates=weeds.covariates,
                    epsvu=c(100,1))
summary(weeds.dspat.5)
AIC(weeds.dspat.5)
coef(weeds.dspat.5)
mu.B = integrate.intensity(weeds.dspat.5,dimyx=120,se=TRUE)
cat('Abundance =        ', round(mu.B$abundance,0), "\n")
cat('Standard Error =  ', round(mu.B$precision$se,0), "\n",
    '95 Percent Conf. Int. =  (', round(mu.B$precision$lcl.95,0), ',',
            round(mu.B$precision$ucl.95,0), ')', '\n')
pdf("NS_model_5_intensity.pdf")
plot(mu.B$lambda, main='Estimated Intensity')
plot(weeds.dspat.5$lines.psp,lty=2,add=TRUE)
plot(owin(poly=weeds.dspat.5$transect),add=TRUE)
plot(weeds.dspat.5$model$Q$data,add=TRUE,pch=20)
dev.off()
# Fit model ~s(x), ~sheep
weeds.dspat.6=dspat(int.formula=~s(x),det.formula=~factor(sheep),
                    study.area=study.area,
                    obs=weeds.obs,lines=weeds.lines,covariates=weeds.covariates,
                    epsvu=c(100,1))

summary(weeds.dspat.6)
AIC(weeds.dspat.6)
coef(weeds.dspat.6)
mu.B = integrate.intensity(weeds.dspat.6,dimyx=120,se=TRUE)
cat('Abundance =        ', round(mu.B$abundance,0), "\n")
cat('Standard Error =  ', round(mu.B$precision$se,0), "\n",
    '95 Percent Conf. Int. =  (', round(mu.B$precision$lcl.95,0), ',',
            round(mu.B$precision$ucl.95,0), ')', '\n')
pdf("NS_model_6_intensity.pdf")
plot(mu.B$lambda, main='Estimated Intensity')
plot(weeds.dspat.6$lines.psp,lty=2,add=TRUE)
plot(owin(poly=weeds.dspat.6$transect),add=TRUE)
plot(weeds.dspat.6$model$Q$data,add=TRUE,pch=20)
dev.off()
# summary of abundance per strip using model 4
Est.N=by(mu.B.4$distribution$N,cut(mu.B.4$distribution$x,seq(0,1200,150)),sum)
True.N=by(weeds.all$x,cut(weeds.all$x,seq(0,1200,150)),length)
postscript("Figure3.ps",height=6,width=5,horizontal=FALSE)
```

```
barplot(rbind(True.N,Est.N),beside=TRUE,legend=TRUE,names.arg=1:8,main="N-S lines model 4")
dev.off()
# Show goodness of fit for sheep absent/present
postscript("Figure4.ps",height=6,width=5,horizontal=FALSE)
exp.nosheep=apply(weeds.dspat.4$exp.counts[1:4,],2,sum)
obs.nosheep=apply(weeds.dspat.4$obs.counts[1:4,],2,sum)
exp.sheep=apply(weeds.dspat.4$exp.counts[5:8,],2,sum)
obs.sheep=apply(weeds.dspat.4$obs.counts[5:8,],2,sum)
par(mfrow=c(2,1))
barplot(rbind(exp=exp.nosheep,obs=obs.nosheep),beside=TRUE,main="Sheep absent")
barplot(rbind(exp=exp.sheep,obs=obs.sheep),beside=TRUE,legend=FALSE,main="Sheep present")
dev.off()
# chi-square test for model 4
chisq=sum((exp.nosheep-obs.nosheep)^2/exp.nosheep)+
sum((exp.sheep-obs.sheep)^2/exp.sheep)
cat("Chi-square=",chisq," p= ",1-pchisq(chisq,2*10-length(weeds.dspat.4$par)),"\n")
# sigma for no sheep and sheep
sigmas=sqrt(1/coef(weeds.dspat.4)$detection)
cat("\n Sigma (no sheep) =",sigmas[1],"\n","Sigma (sheep)     =",sigmas[2],"\n")
###############################################################################
# Modify sampled vertical N-S strips to extend from 600 to 1200 and then
# add 4 E-W horizontal strips centered at 75,225,375,525. Using approximate
# detection functions for sheep/no sheep areas, a sample of observations from
# the points are randomly selected.
#
# NOTE: The following is random and will not produce the same results each time
# it is run because of the random observation process.
#
###############################################################################
data(weeds.obs)
data(weeds.lines)
weeds.obs=weeds.obs[weeds.obs$y>600,]
xlines=data.frame(label=9:12,x0=rep(0,4),x1=rep(1200,4),y0=c(75,225,375,525),
                  y1=c(75,225,375,525),width=rep(149.999,4))
ls=lines_to_strips(xlines,study.area)
pts=ppp(x=weeds.all$x,y=weeds.all$y,window=study.area)
pdf("E-W_N-S samples.pdf")
plot(pts)
plot(ppp(x=weeds.obs$x,y=weeds.obs$y,window=study.area),add=TRUE,pch=19,col="red",cex=.5)
obs=sample.points(ls$transects,xlines,pts,detfct=hndetfct,
                  det.par=c(3.637586,-.1466),det.formula=~factor(sheep),
                  covariates=weeds.covariates)
weeds.obs=rbind(weeds.obs,obs)
plot(ppp(x=obs$x,y=obs$y,window=study.area),add=TRUE,pch=19,cex=.5)
dev.off()
weeds.lines[,"y0"]=600.0001
weeds.lines=rbind(weeds.lines,as.matrix(xlines))
weeds.dspat=dspat(int.formula=~factor(strip),det.formula=~factor(sheep),
                  study.area=study.area,
                  obs=weeds.obs,lines=weeds.lines,covariates=weeds.covariates,
                  epsvu=c(100,1),nclass=15)
coef(weeds.dspat)
# sigma for no sheep and sheep
```

```
sigmas=sqrt(1/coef(weeds.dspat)$detection)
cat("\n Sigma (no sheep) =",sigmas[1],"\n","Sigma (sheep)     =",sigmas[2],"\n")
mu.B <- integrate.intensity(weeds.dspat,dimyx=120,se=TRUE)
cat('Abundance =        ', round(mu.B$abundance,0), "\n")
cat('Standard Error =  ', round(mu.B$precision$se,0), "\n",
    '95 Percent Conf. Int. =   (', round(mu.B$precision$lcl.95,0), ',',
           round(mu.B$precision$ucl.95,0), ')', '\n')
pdf("E-W_N-S Estimated Intensity.pdf")
plot(mu.B$lambda, main='Estimated Intensity')
plot(weeds.dspat$lines.psp,lty=2,add=TRUE)
plot(owin(poly=weeds.dspat$transect),add=TRUE)
plot(weeds.dspat$model$Q$data,add=TRUE,pch=20)
dev.off()
# summary of abundance per strip
pdf("E-W_N-S AbundanceByStrip.pdf")
Est.N=by(mu.B$distribution$N,cut(mu.B$distribution$x,seq(0,1200,150)),sum)
True.N=by(weeds.all$x,cut(weeds.all$x,seq(0,1200,150)),length)
barplot(rbind(True.N,Est.N),beside=TRUE,legend=TRUE,names.arg=1:8,main="N-S and E-W lines")
dev.off()
# Show goodness of fit for sheep absent/present
pdf("GOF for NS_EW model.pdf")
exp.nosheep=apply(weeds.dspat$exp.counts[1:4,],2,sum)
obs.nosheep=apply(weeds.dspat$obs.counts[1:4,],2,sum)
exp.sheep=apply(weeds.dspat$exp.counts[5:8,],2,sum)
obs.sheep=apply(weeds.dspat$obs.counts[5:8,],2,sum)
par(mfrow=c(2,1))
barplot(rbind(exp=exp.nosheep,obs=obs.nosheep),beside=TRUE,legend=TRUE,main="Sheep absent")
barplot(rbind(exp=exp.sheep,obs=obs.sheep),beside=TRUE,legend=FALSE,main="Sheep present")
dev.off()
# chi-square test for model
chisq=sum((exp.nosheep-obs.nosheep)^2/exp.nosheep)+
sum((exp.sheep-obs.sheep)^2/exp.sheep)
cat("Chi-square=",chisq," p= ",1-pchisq(chisq,2*15-10),"\n")


###########################################################################
# Modify sampling such that all strips are E-W. Using approximate
# detection functions for sheep/no sheep areas derived from known data,
# a sample of observations from the points are randomly selected.
#
# NOTE: The following is random and will not produce the same results each time
# it is run because of the random observation process.
#
###########################################################################
xlines=data.frame(label=1:8,x0=rep(0,8),x1=rep(1200,8),y0=seq(75,1125,150),y1=seq(75,1125,150),
                       width=rep(149.999,8))
ls=lines_to_strips(xlines,study.area)
pts=ppp(x=weeds.all$x,y=weeds.all$y,window=study.area)
pdf("E-W samples.pdf")
plot(pts)
obs=sample.points(ls$transects,xlines,pts,detfct=hndetfct,
                 det.par=c(3.637586,-.1466),det.formula=~factor(sheep),
                 covariates=weeds.covariates)
plot(ppp(x=obs$x,y=obs$y,window=study.area),add=TRUE,pch=19,cex=.5)
```

```
dev.off()
weeds.dspat=dspat(int.formula=~factor(strip),det.formula=~factor(sheep),
                      study.area=study.area,
                      obs=obs,lines=xlines,covariates=weeds.covariates,
                      epsvu=c(100,1),nclass=15)
coef(weeds.dspat)
sigmas=sqrt(1/coef(weeds.dspat)$detection)
cat("\n Sigma (no sheep) =",sigmas[1],"\n","Sigma (sheep)    =",sigmas[2],"\n")
mu.B <- integrate.intensity(weeds.dspat,dimyx=120,se=TRUE)
cat('Abundance =        ', round(mu.B$abundance,0), "\n")
cat('Standard Error =  ', round(mu.B$precision$se,0), "\n",
    '95 Percent Conf. Int. =  (', round(mu.B$precision$lcl.95,0), ',',
           round(mu.B$precision$ucl.95,0), ')', '\n')
pdf("E-W Estimated Intensity.pdf")
plot(mu.B$lambda, main='Estimated Intensity')
plot(weeds.dspat$lines.psp,lty=2,add=TRUE)
plot(owin(poly=weeds.dspat$transect),add=TRUE)
plot(weeds.dspat$model$Q$data,add=TRUE,pch=20)
dev.off()
# summary of abundance per strip
Est.N=by(mu.B$distribution$N,cut(mu.B$distribution$x,seq(0,1200,150)),sum)
True.N=by(weeds.all$x,cut(weeds.all$x,seq(0,1200,150)),length)
pdf("E-W AbundanceByStrip.pdf")
barplot(rbind(True.N,Est.N),beside=TRUE,legend=TRUE,names.arg=1:8,main="E-W lines")
dev.off()
# Show goodness of fit for sheep absent/present
pdf("GOF for EW model.pdf")
exp.nosheep=apply(weeds.dspat$exp.counts[1:4,],2,sum)
obs.nosheep=apply(weeds.dspat$obs.counts[1:4,],2,sum)
exp.sheep=apply(weeds.dspat$exp.counts[5:8,],2,sum)
obs.sheep=apply(weeds.dspat$obs.counts[5:8,],2,sum)
par(mfrow=c(2,1))
barplot(rbind(exp=exp.nosheep,obs=obs.nosheep),beside=TRUE,legend=TRUE,main="Sheep absent")
barplot(rbind(exp=exp.sheep,obs=obs.sheep),beside=TRUE,legend=FALSE,main="Sheep present")
dev.off()
# chi-square test for model
chisq=sum((exp.nosheep-obs.nosheep)^2/exp.nosheep)+
sum((exp.sheep-obs.sheep)^2/exp.sheep)
cat("Chi-square=",chisq," p= ",1-pchisq(chisq,2*15-10),"\n")
}
```

---

weeds.all   *Dubbo weed data with constructed y-coordinate*

Description

Locations of devils claw in a farming paddock. Locations to all weeds are given as x,y coordinates and are contained in one of eight 150m wide transects (75m each side). The weeds seen by observers are specified as Seen=1.

Usage

```
data(weeds.all)
```

Format

A data frame with 742 observations on the following 5 variables.

label label of the transect 1 to 8

x x coordinate along horizontal (east-west) of 1200m x 1200m paddock

y y coordinate along vertical (north-south) of 1200m x 1200m paddock

distance absolute perpendicular distance from line

Seen weed was seen if 1 and 0 if missed

Details

The data provided from Melville and Welsh did not have the y-coordinate. We have constructed y-coordinates by drawing randomly from a uniform distribution in the y-direction such that no two weeds are at the exact same location. The code used to create weeds.all was as follows:

```
data(weeds)
# Fudge the data ever so slightly to appease spatstat so the transects don't abut
# and all points are contained within the strips.
weeds$SignedDistance[weeds$SignedDistance==75]=74.99
weeds$SignedDistance[weeds$SignedDistance==-75]=-74.99
weeds.all=data.frame(label=weeds$Transect, x=(weeds$Transect-1)*150+75
                +weeds$SignedDistance, y=floor(runif(dim(weeds)[1])*1200),
                distance=weeds$Distance, Seen=weeds$Seen)
while(any(duplicated(data.frame(x=weeds.all$x,y=weeds.all$y))))
{
  npts=sum(as.numeric(any(duplicated(data.frame(x=weeds.all$x,y=weeds.all$y)))))
   weeds.all$y[duplicated(data.frame(x=weeds.all$x,y=weeds.all$y))]=
                                                            runif(npts)*1200
}
save(weeds.all,file="weeds.all.rda")
```

See weeds for more details.

References

Melville, G. J., and A. H. Welsh. 2001. Line transect sampling in small regions. Biometrics 57:1130-1137.

weeds.covariates	*Covariate grid for Dubbo weed data*

Description

Grid (1 sq meter) of covariates for farm paddock in Dubbo weed data.

Usage

```
data(weeds.covariates)
```

Format

A data frame with 120 x 120 observations on the following 4 variables.

x x coordinate for mid-point of grid cell

y y coordinate for mid-point of grid cell

sheep 0 if no sheep and 1 if sheep were present on the transect

strip transect number 1 to 8

Details

This is the constructed set of covariates for the farm paddock for the weeds data that were provided by Melville and Welsh (see reference below) that were used in the Biometrics paper on distance sampling.

The code used to create the covariate grid was as follows:

```
xx=expand.grid(seq(5,1195,10),seq(5,1195,10))
weeds.covariates=data.frame(x=xx$Var2,y=xx$Var1,
                 sheep=rep(c(0,1),each= 120^2/2),
                 strip=rep(c(1,2,3,4,5,6,7,8),each=120^2/8))
save(weeds.covariates,file="weeds.covariates.rda")
```

References

Melville, G. J., and A. H. Welsh. 2001. Line transect sampling in small regions. Biometrics 57:1130-1137.

---

weeds.lines   *Transect lines from Dubbo weed data*

Description

Lines sampled in a farming paddock with eight 150m wide transects (75m each side)

Usage

```
data(weeds.lines)
```

Format

A data frame with 8 observations on the following 6 variables.

label Label of the transect 1 to 8

x0 x coordinate for the beginning of the line

x1 x coordinate for the end of the line

y0 y coordinate for the beginning of the line

y1 y coordinate for the end of the line

width full width of the transect

Details

These are lines constructed for the weeds data that were provided by Melville and Welsh (see reference below) that were used in the Biometrics paper on distance sampling.

The code used to create the lines was as follows:

```
weeds.lines=data.frame(label=1:8,x0=75+0:7*150,x1=75+0:7*150,
    y0=rep(0,8),y1=rep(1200,8),width=rep(149.9999,8))
```

The line widths were reduced by 0.0001 so the transects do no abut because spatstat treats them as overlapping polygons.

References

Melville, G. J., and A. H. Welsh. 2001. Line transect sampling in small regions. Biometrics 57:1130-1137.

weeds.obs	*Observations from Dubbo weed data*

Description

Observed devils claw in a farming paddock from eight 150m wide transects (75m each side). These are the records from weeds that were seen.

Usage

```
data(weeds.obs)
```

Format

A data frame with 479 observations on the following 4 variables.

label Label of the transect 1 to 8

x x coordinate in the farming paddock

y y coordinate created randomly for the data

distance perpendicular distance from line to weed

Details

These are the data constructed from weeds that were provided by Melville and Welsh (see reference below) that were used in the Biometrics paper on distance sampling.

The code used to create the data from weeds was as follows:

```
data(weeds.all)
weeds.obs=weeds.all[weeds.all$Seen==1,]
weeds.obs$Seen=NULL
save(weeds.obs,file="weeds.obs.rda")
```

References

Melville, G. J., and A. H. Welsh. 2001. Line transect sampling in small regions. Biometrics 57:1130-1137.

Index