Package 'DSsim'

February 13, 2019

Depends mrds, methods

Imports graphics, splancs, mgcv, shapefiles, rgeos, fields, sp

Suggests testthat, parallel, pbapply, knitr, rmarkdown

VignetteBuilder knitr

Type Package

Title Distance Sampling Simulations

Version 1.1.4

LazyLoad yes

Author Laura Marshall <lhm@st-and.ac.uk>

Maintainer Laura Marshall <lhm@st-and.ac.uk>

Description Performs distance sampling simulations.It repeatedly generates instances of a user defined population within a given survey region, generates realisations of a survey design (currently these must be generated using Distance software in advance <http://distancesampling.org/>) and simulates the detection process. The data are then analysed so that the results can be compared for accuracy and precision across all replications. This will allow users to select survey designs which will give them the best accuracy and precision given their expectations about population distribution. Any uncertainty in population distribution or population parameters can be included by running the different survey designs for a number of different population descriptions. An example simulation can be found in the help file for make.simulation.

License GPL (>= 2)

Collate 'DDF.Data.R' 'generic.functions.R' 'DDF.Analysis.R' 'DS.Analysis.R' 'Survey.Design.R' 'PT.Design.R' 'PT.Nested.Design.R' 'PT.Systematic.Design.R' 'PT.Random.Design.R' 'LT.Design.R' 'LT.User.Specified.Design.R' 'LT.EqSpace.ZZ.Design.R' 'LT.EqAngle.ZZ.Design.R' 'LT.Random.Design.R' 'LT.Systematic.Design.R' 'Density.R' 'Population.Description.R' 'Region.R' 'Region.Table.R' 'Sample.Table.R' 'Obs.Table.R' 'Single.Obs.DDF.Data.R' 'Transect.R' 'Line.Transect.R' 'Detectability.R' 'Population.R'

'LT.Survey.Results.R' 'Survey.Results.R' 'DSM.Analysis.R'
'Simulation.R' 'Class.Constructors.R' 'Design.Summary.R'
'LT.SegmentedGrid.Design.R' 'LT.SegmentedTrack.Design.R'
'Survey.R' 'LT.Survey.R' 'Point.Transect.R' 'PT.Survey.R'
'Population.Summary.R' 'Simulation.Summary.R'
'Single.Obs.LT.Survey.R' 'Single.Obs.PT.Survey.R'
'accumulate.PP.results.R' 'accumulate.warnings.R'
'add.covariate.values.R' 'add.dist.error.R' 'add.miss.dists.R'
'add.summary.results.R' 'calc.area.R'
'calc.poss.detect.dists.lines.R'
'calc.poss.detect.dists.lines.largeN.R'
'calc.poss.detect.dists.points.R' 'calculate.fitted.R'
'calculate.scale.param.R' 'check.LinkID.order.R'
'check.covariates.R' 'check.intersection.R' 'check.shapefile.R'
'check.sim.setup.R' 'coords.from.shapefile.R' 'create.bins.R'
'create.results.arrays.R' 'data.for.distance.R'
'description.summary.R' 'dssim.update.R'
'extract.spat.poly.coords.R' 'generate.pop.D.R'
'generate.pop.N.R' 'get.ave.density.R' 'get.bound.box.R'
'get.line.sampler.info.R' 'get.point.sampler.info.R'
'get.shapefile.names.R' 'get.surface.constant.R'
'get.surface.gam.R' 'hn.detect.R' 'hr.detect.R' 'in.polygons.R'
'is.gap.R' 'message.handler.R' 'modify.strata.for.analysis.R'
'rename.duplicates.R' 'rtpois.R' 'save.sim.results.R'
'setcov.R' 'simulate.detections.R' 'single.simulation.loop.R'
'store.ddf.results.R' 'store.dht.results.R'

RoxygenNote 6.1.1

NeedsCompilation no

Repository CRAN

Date/Publication 2019-02-13 12:30:03 UTC

R topics documented:

`add.hotspot` *S4 generic method to add a hotspot to the density grid*

Description

Uses a Gaussian decay around a central location to add a hotspot to the density grid.

Usage

```
add.hotspot(object, centre, sigma, amplitude)

## S4 method for signature 'Density'
add.hotspot(object, centre, sigma, amplitude)
```

Arguments

`object`	an object of class Density or Simulation
`centre`	an x,y-coordinate giving the centre of the hotspot
`sigma`	a value giving the scale parameter for a gaussian decay
`amplitude`	the height of the hotspot at its centre

Value

the updated Density or Simulation object

See Also

`make.density`

check.sim.setup *check.sim.setup*

Description

A function which allows the user to check the simulation setup. It displays a panel of 4 diagnostic plots: top left - study region with example population, top right - study region with transects, bottom left - an example realisation of a survey with detected animals shown in cyan and undetected animals shown in red, bottom right - a histogram of the example distances to which the detection function would be fitted.

Usage

```
check.sim.setup(simulation)
```

Arguments

simulation	A simulation object created by a call to make.simulation

Value

a invisible copy of the simulation

Author(s)

Eric Rexstad, Laura Marshall

Examples

```
## Not run:
sim <- make.simulation(design.obj = make.design("point"))
check.sim.setup(sim)

## End(Not run)
```

cov.summary.list *Covariate Truncation Simulation Summaries*

Description

This is a list of 5 simulation summaries. Data were generated using a systematic line transect design and a mixture of two half normal detection functions describing detectability, one for males and one for females. Each of the five simulation summaries corresponds to a different analysis truncation distance when fitting the detection function model which was selected from either a half normal or hazard rate based on the minimum AIC. These simulations investigate whether truncation distance affects the accuracy and precision of the estimates of abundance / density.

Usage

```
data("trunc_cov_summary")
```

Format

The format is: List of 5

`t200` a simulation summary (truncation = 200)

`t400` a simulation summary (truncation = 400)

`t600` a simulation summary (truncation = 600)

`t800` a simulation summary (truncation = 800)

`t1000` a simulation summary (truncation = 1000)

Examples

```
data(trunc_cov_summary)
cov.summary.list$t200
```

`covmod.summary.list` *Covariate Truncation Simulation Summaries*

Description

This is a list of 5 simulation summaries. Data were generated using a systematic line transect design and a mixture of two half normal detection functions describing detectability, one for males and one for females. Each of the five simulation summaries corresponds to a different analysis truncation distance when fitting the detection function model in which detectability was modelled as a function of sex. These simulations investigate whether truncation distance affects the accuracy and precision of the estimates of abundance / density.

Usage

```
data("covmod_summary")
```

Format

The format is: List of 5

`t200` a simulation summary (truncation = 200)

`t400` a simulation summary (truncation = 400)

`t600` a simulation summary (truncation = 600)

`t800` a simulation summary (truncation = 800)

`t1000` a simulation summary (truncation = 1000)

Examples

```
data(covmod_summary)
covmod.summary.list$t200
```

create.bins *Create bins from a set of binned distances and a set of cutpoints.*

Description

This is a service routine and shouldn't be necessary in normal analyses.

Usage

```
create.bins(data, cutpoints)
```

Arguments

data	data.frame with at least the column distance.
cutpoints	vector of cutpoints for the bins

Value

data data with two extra columns distbegin and distend.

Author(s)

David L. Miller

create.region.table *S4 generic method to generate a region table*

Description

This function is called internally to generate a region table required to estimate abundance / density via the Hortvitz-Thompson estimator.

Usage

```
create.region.table(object, region)

## S4 method for signature 'Survey'
create.region.table(object, region)
```

Arguments

object	an object of a class inheriting from Survey
region	an object of class Region

Value

an object of class Region.Table

create.sample.table *S4 generic method to generate a sample table*

Description

This function is called internally to generates a sample table required to estimate abundance / density via the Hortvitz-Thompson estimator.

Usage

```
create.sample.table(object)

## S4 method for signature 'Survey'
create.sample.table(object)
```

Arguments

object	an object of a class inheriting from Survey

Value

an object of class Sample.Table

create.survey.results *S4 generic method to simulate a survey*

Description

Simulates the process by which individuals / clusters are detected. Currently this is only implemented for line transect surveys. It returns an object of class LT.Survey.Results which contains a population, a set of transects, distance data and if requested region, sample and obs tables.

Usage

```
create.survey.results(object, dht.tables = FALSE, ...)

## S4 method for signature 'Simulation'
create.survey.results(object, dht.tables = FALSE,
  ...)

## S4 method for signature 'Single.Obs.LT.Survey'
create.survey.results(object,
  dht.tables = FALSE, ...)

## S4 method for signature 'Single.Obs.PT.Survey'
create.survey.results(object,
  dht.tables = FALSE, ...)
```

Arguments

object	an object of class Simulation
dht.tables	logical value indicating whether or the data tables for Hortvitz-Thompson estimation are required.
...	allows a region object to be passed in

Details

This object can be displayed using `plot()` or the distance data extracted using `get.distance.data()`. You can then investigate fitting models to this data.

Value

an object of class LT.Survey.Results

Examples

```
## Not run:
survey.results <- create.survey.results(simulation, dht.table = TRUE)

plot(survey.results)

## End(Not run)
```

---

data.for.distance *data.for.distance*

Description

Formats the data generated by create.survey.results into a form suitable for import into Distance

Usage

```
data.for.distance(object, file = NULL, round = 2, sep = "\t")
```

Arguments

object	an object of class Survey.Results generated by create.survey.results
file	path and filename if the user would like the results saved to file. Should contain the .txt file extension.
round	the amount of decimal places to round the distances to
sep	the field separator string for writing to file

Author(s)

L Marshall

DDF.Analysis-class *Class "DDF.Analysis"*

Description

Class "DDF.Analysis" is an S4 class describing a detection function which is to be fitted to the data.

Slots

`dsmodel` Object of class "formula"; describing the detection function model.

`criteria` Object of class "character"; describes which model selection criteria to use ("AIC","AICc","BIC").

`truncation` Object of class "character"; Specifies the truncation distance for the analyses.

`binned.data` Object of class "character"; logical value specifying if the data should be binned for analysis.

`cutpoints` Object of class "character"; gives the cutpoints of the bins for binned data analysis.

`analysis.strata` Dataframe with two columns ("design.id" and "analysis.id"). The former gives the strata names as defined in the design (i.e. the region object) the second specifies how they should be grouped (into less strata) for the analyses

`ddf.result` Object of class "list"; object of S3 class ddf.

Methods

`run.analysis signature=c(object = "DDF.Analysis", data = "DDF.Data")`: runs the analysis described in the object on the data provided.

See Also

`make.ddf.analysis.list`

DDF.Data-class *S4 Class "DDF.Data"*

Description

Class "DDF.Data"

Details

A virtual class containing a data.frame with distance sampling data in the correct format for mrds.

Slots

`ddf.dat` Object of class "data.frame"; dataframe with all the necessary column to fit a detection function using mrds.

Density-class *Class "Density"*

Description

Class "Density" is an S4 class containing a list of grids which describe the density of individuals / clusters of a population. The list contains one grid (data.frame) for each strata.

Slots

region.name Object of class "character"; the region name.

strata.name Object of class "character"; the strata names

density.surface Object of class "list"; list of data.frames with the columns x, y and density. There must be one data.frame for each strata.

x.space Object of class "numeric"; The spacing between gridpoints described in the density data.frames in the x-direction.

y.space Object of class "numeric"; The spacing between gridpoints described in the density data.frames in the y-direction.

units Object of class "numeric"; The units of the grid points.

See Also

make.density

description.summary *Provides a description of the summary object/output Prints a list of the terms used in the simulation summary.*

Description

Provides a description of the summary object/output

Prints a list of the terms used in the simulation summary.

Usage

```
description.summary()
```

Author(s)

Laura Marshall

Design.Summary-class *S4 Class "Design.Summary"*

Description

Class "Design.Summary"

Details

Class "Design.Summary" is an S4 class containing a summary of the survey design. This is returned when summary(Design) is called. If it is not assigned to a variable the object will be displayed via the show method.

Methods

show signature=(object = "Design.Summary"): prints the contents of the object in a user friendly format.

Detectability-class *S4 Class "Detectability"*

Description

Class "Detectability"

Details

Class "Detectability" is an S4 class describing the probability of detecting individuals / clusters in a population.

Slots

key.function Object of class "character"; a code specifying the detection function form ("hn" = half normal, "hr" = hazard rate.)

scale.param Object of class "numeric"; The scale parameter for the detection function.

shape.param Object of class "numeric"; The shape parameter for the detection function.

cov.param Object of class "numeric"; The parameter values associated with the covariates. Not yet implemented

truncation Object of class "numeric"; The maximum distance at which objects may be detected.

See Also

make.detectability

generate.population *S4 generic method to generate an instance of a population*

Description

Uses the population description and detectability details to generate an instance of the population. Note that if the first argument supplied is of class Population.Description rather than class Simulation then a second argument `detectabity` must also be supplied and must be an object of class Detectability.

Usage

```
generate.population(object, ...)

## S4 method for signature 'Population.Description'
generate.population(object,
  detectability, region.obj = NULL)

## S4 method for signature 'Simulation'
generate.population(object, ...)
```

Arguments

`object`	an object of class Simulation or Population.Description
`...`	when this is called on an object of class Population.Description the additional arguments detectability and region.obj should also be supplied
`detectability`	object of class Detectability (optional - only required if object is of class Population.Description)
`region.obj`	the region object for the population (optional - only required if object is of class Population.Description)

Value

an object of class Population

generate.transects *S4 generic method to generate an instance of a design*

Description

Uses the Survey.Design details to generate transects. Currently this involves loading a survey shapefile from the path specified in the Survey.Design object and can only work with line transect designs.

Usage

```
generate.transects(object, region = NULL, ...)

## S4 method for signature 'PT.Design'
generate.transects(object, region = NULL,
  index = NULL)

## S4 method for signature 'PT.Nested.Design'
generate.transects(object, region = NULL,
  index = NULL, silent = FALSE)

## S4 method for signature 'PT.Systematic.Design'
generate.transects(object,
  region = NULL, index = NULL, silent = FALSE)

## S4 method for signature 'LT.Design'
generate.transects(object, region = NULL,
  index = NULL)

## S4 method for signature 'LT.EqSpace.ZZ.Design'
generate.transects(object,
  region = NULL, index = NULL, silent = FALSE, complement = FALSE)

## S4 method for signature 'LT.Systematic.Design'
generate.transects(object,
  region = NULL, index = NULL, silent = FALSE)

## S4 method for signature 'Simulation'
generate.transects(object, region = NULL)
```

Arguments

object	an object of class Simulation or a class which inherits from Survey.Design
region	optional only required if object is of class Survey.Design.
...	optional argument index if an object of class Survey.Design is supplied allowing the user to access / plot different sets of transects listed in the filenames slot.
index	specifies which set of transect should be loaded
silent	if TRUE does not report warnings about a single value for nested spacing with a multi strata region
complement	logical indicating whether two sets of complimentary transects should be generated

Value

an object of class Line.Transect

get.area *Returns the area of the region*

Description

Returns the area of the region

Usage

```
get.area(object)

## S4 method for signature 'Region'
get.area(object)
```

Arguments

object	object of class Region

Value

numeric value specifying the area of the region

get.distance.data *S4 generic method to extract distance data*

Description

Extracts distance data from a Survey.Results object

Usage

```
get.distance.data(object)

## S4 method for signature 'LT.Survey.Results'
get.distance.data(object)

## S4 method for signature 'Survey.Results'
get.distance.data(object)
```

Arguments

object	an object of class LT.Survey.Results

Value

a data.frame describing the distance data

See Also

create.survey.results

`get.N` *S4 generic method to return N*

Description

Returns the population size

Usage

```
get.N(object)

## S4 method for signature 'Population.Description'
get.N(object)
```

Arguments

`object`	an object of class Population.Description

Value

numeric value of the population size

`histogram.N.ests` *histogram.N.ests*

Description

Plots a histogram of the estimates abundances

Usage

```
histogram.N.ests(x, ...)
```

Arguments

`x`	object of class Simulation
`...`	optional parameters to pass to the generic hist function in graphics

Line.Transect-class *S4 Class "Line.Transect"*

Description

Class "Line.Transect" contains an instance of a Line Transect Survey

Methods

`plot signature=(object = "Line.Transect")`: plots the transects.

LT.Design-class *Virtual Class "LT.Design" extends Class "Survey.Design"*

Description

Virtual Class `"LT.Design"` is an S4 class detailing the type of line transect design.

Methods

`generate.transects signature=(object = "LT.Design", ...)`: loads a set of transects from a shapefile.

See Also

`make.design`

LT.Survey-class *Virtual Class "LT.Survey" extends class "Survey"*

Description

Virtual Class `"LT.Survey"` is an S4 class containing a population and a set of transects.

Slots

`perpendicular.truncation` Object of class `"numeric"`; the maximum distance from the transect at which animals may be detected.

See Also

`make.design`

LT.Survey.Results-class

S4 Class "LT.Survey.Results"

Description

Class containing all the components relating to a single realisation of a survey.

Slots

`region` Object of class `"Region"`; the region representation.

`population` Object of class `"Population"`; the population.

`transects` Object of class `"Line.Transect"`; the transects.

`ddf.data` Object of class `"Single.Obs.DDF.Data"`; The ddf data for `ddf`. @slot obs.table Object of class `"Obs.Table"`; One of the tables for `dht`. @slot sample.table Object of class `"Sample.Table"`; One of the tables for `dht`. @slot region.table Object of class `"Region.Table"`; One of the tables for `dht`.

Methods

`plot signature=(object = "LT.Survey.Results")`: plots the region, the location of individuals in the population, the transects and the successful sightings.

`get.distance.data signature=(object = "LT.Survey.Results")`: returns the ddf data as a dataframe..

make.ddf.analysis.list

Creates a list of DDF.Analysis objects

Description

This method creates a list of DDF.Analysis objects each of which describes a model to fit to the distance data. The simulation will fit each of these models to the data generated in the simulation and select the model with the minimum criteria value.

Usage

```
make.ddf.analysis.list(dsmodel = list(~cds(key = "hn", formula = ~1)),
  mrmodel = NULL, method = "ds", criteria = "AIC",
  analysis.strata = data.frame(), truncation = 50,
  binned.data = FALSE, cutpoints = numeric(0))
```

Arguments

dsmodel	list of distance sampling model formula specifying the detection function (see ?ddf for further details)
mrmodel	not yet implemented
method	character only "ds" normal distance sampling currently implemented
criteria	character model selection criteria (AIC, AICc, BIC) - only AIC implemented at present.
analysis.strata	Dataframe with two columns ("design.id" and "analysis.id"). The former gives the strata names as defined in the design (i.e. the region object) the second specifies how they should be grouped (into less strata) for the analyses
truncation	numeric truncation distance for analyses
binned.data	logical whether the data should be analsed in bins
cutpoints	gives the cutpoints of the binned data

Details

By default this function creates a half-normal detection function model `dsmodel = list(~cds(key = "hn", formula = ~1)` with a truncation distance of 75.

Value

list of objects of class DDF.Analysis

Author(s)

Laura Marshall

See Also

`ddf` in `library(mrds)`

Examples

```
# A simple half-normal "ds" model can be created using the default values
ddf.analyses <- make.ddf.analysis.list()

# To incorporate model selection between a 'hn' and 'hr' model:
ddf.analyses <- make.ddf.analysis.list(dsmodel = list(~cds(key = "hn",
 formula = ~1),~cds(key = "hr", formula = ~1)), method = "ds",
 criteria = "AIC")
```

`make.density` *Creates a Density object*

Description

The user has the option to create a grid describing the density of the objects and pass this in giving the x and y spacings used in the creation of this grid. Alternatively the user can specify a constant density and x, y spacings and this grid will be generated automatically. The user may also supply a `mgcv gam` object and x, y spacings and the density grid will be created from these.

Usage

```
make.density(region.obj = make.region(), density.surface = list(),
  x.space = 5, y.space = NULL, buffer = numeric(0),
  constant = numeric(0), density.gam = NULL, dsm = NULL,
  formula = NULL)
```

Arguments

`region.obj`	the Region object in which the density grid will be created
`density.surface`	Object of class `list`; list of data.frames with the columns x, y and density. There must be one data.frame for each strata.
`x.space`	the intervals in the grid in the x direction
`y.space`	the intervals in the grid in the y direction
`buffer`	the width of the buffer region for generating the density grid. If not supplied DSsim will use the maximum value provided for the x.space or y.space.
`constant`	a value describing a constant density across the surface. If not supplied a default value of 1 is used for all strata.
`density.gam`	`gam` object created using `mgcv` with only x and y as explanatory covariates.
`dsm`	not currently implemented
`formula`	not currently implemented

Value

object of class Density

Author(s)

Laura Marshall

See Also

`make.region`

Examples

```
# A simple density surface with a constant value of 1 can be created within a rectangular
# region using
# the default values:
density <- make.density()
plot(density)
plot(make.region(), add = TRUE)

# The example below shows hot to add high and low point to the density surface
## Not run:
pop.density <- make.density(region.obj = region, x.space = 10,
 y.space = 10, constant = 0.5)

pop.density <- add.hotspot(pop.density, centre = c(50, 200),
 sigma = 100, amplitude = 0.1)
pop.density <- add.hotspot(pop.density, centre = c(500, 700),
 sigma = 900, amplitude = 0.05)
pop.density <- add.hotspot(pop.density, centre = c(300, 100),
 sigma = 100, amplitude = -0.15)

#New plot features
plot(pop.density)
plot(region, add = TRUE)

#Block style plotting
plot(pop.density, contours = FALSE, style = "blocks")
plot(region, add = TRUE)


## End(Not run)
```

---

make.design *Creates a Survey.Design object*

Description

Currently surveys are only generated within the GIS in Distance. If you are running a simulation in R you will need to get Distance to generate all the surveys as shapefiles in advance and supply the path to the directory which contains these shapefiles and only these shapefiles.

Usage

```
make.design(transect.type = "line", design.details = "default",
  region.obj = "region", design.axis = 0, spacing = 100,
  nested.space = numeric(0), no.complex = numeric(0),
  angle = numeric(0), plus.sampling = logical(0),
  path = character(0))
```

Arguments

transect.type	character variable specifying either "Line" or "Point"
design.details	a character vector describing the type of design. See details section.
region.obj	the character name of the Region object where the survey is to be carried out.
design.axis	user may provide the angle of the design axis but not currently used
spacing	user may provide the systematic design spacing but but not currently used
nested.space	the number of spaces between nested points. If spacing = 1 then all points on the systematic design will be selected.
no.complex	the number of complex detectors to distribute based on simple random sampling of the systematic grid of detectors.
angle	user may provide the design angle (only relevant in equal angle zigzag designs) but not currently used
plus.sampling	logical value indicating whether a plus sampling protocol is used but not currently used
path	pathway giving the location of the folder of survey shapefiles

Details

The design.details argument should specify a character vector of either 1 or 2 elements. These options are described in the table below:

Transect Type	Design Details	
Line	Parallel	Systematic
Line	Parallel	Random
Line	Zigzag	Equal Angle
Line	Zigzag	Equal Spaced
Line	User Specified	
Point	Systematic	
Point	Random	
Point	Nested	

Value

object of a class which inherits from class Survey.Design

Author(s)

Laura Marshall

Examples

```
# DSsim can generate a systematic set of parallel line transects which by default have a
# spacing of 100
design <- make.design("line")
```

```
# The easiest way to generate the transect is by creating a simulation (default simulations
#create a line transect design)
sim <- make.simulation()
transects <- generate.transects(sim)
plot(make.region())
plot(transects, col = 4, lwd = 2)

# DSsim can generate a systematic grid of point transects which by default have a spacing of 100
design <- make.design("point")

sim <- make.simulation(design.obj = design)
transects <- generate.transects(sim)
plot(make.region())
plot(transects)

# More complex designs can be defined in Distance for Windows. This software can then generate
# multiple survey instances and store them as shapefiles for use by DSsim. The shapefile below
# was generated in this way.

## Not run:

coords <- gaps <- list()
coords[[1]] <- list(data.frame(x = c(0,1000,1000,0,0), y = c(0,0,
 1000,1000,0)))
gaps[[1]] <- list(data.frame(x = c(400,600,500,350,400), y = c(100,
 250,600,120,100)))
region <- make.region(region.name = "study.area", units = "m",
 coords = coords, gaps = gaps)

data(transects.shp)
#Edit the pathway below to point to an empty folder where the
#transect shapefile will be saved
shapefile.pathway <- "C:/..."
library(shapefiles)
write.shapefile(transects.shp, paste(shapefile.pathway,"/transects_1",
 sep = ""))

# This design was created in Distance for Windows in a region with the same dimensions as the
# deault make.region().
parallel.design <- make.design(transect.type = "Line",
 design.details = c("Parallel","Systematic"), region = region,
 design.axis = 0, spacing = 100, plus.sampling =FALSE,
 path = shapefile.pathway)

# As there is only one set of transects we have to set single.transect.set = TRUE
sim <- make.simulation(single.transect.set = TRUE, design.obj = parallel.design)
transects <- generate.transects(sim)
plot(region)
plot(transects, col = 4, lwd = 2)

## End(Not run)
```

`make.detectability` *Creates a Detectablility object*

Description

The detectability of the population is described by the values in this class.

Usage

```
make.detectability(key.function = "hn", scale.param = 25,
  shape.param = numeric(0), cov.param = list(), truncation = 50)
```

Arguments

`key.function`	specifies shape of the detection function (either half-normal "hn", hazard rate "hr" or uniform "uf")
`scale.param`	numeric vector with either a single value to be applied globally or a value for each strata. These should be supplied on the natural scale.
`shape.param`	numeric vector with either a single value to be applied globally or a value for each strata. These should be supplied on the natural scale.
`cov.param`	Named list with one named entry per individual level covariate. Covariate parameter values should be defined on the log scale (rather than the natural scale), this is the same scale as provided in the ddf output in mrds and also in the MCDS output in Distance. Cluster sizes parameter values can be defined here. Each list entry will either be a data.frame containing 2 or 3 columns: level, param and where desired strata. If the region has multiple strata but this column is omitted then the values will be assumed to apply globally. The cluster size entry in the list must be named 'size'. Alternatively the list element may a numeric vector with either a single value to be applied globally or a value for each strata.
`truncation`	the maximum perpendicular (or radial) distance at which objects may be detected from a line (or point) transect.

Value

object of class Detectablility

Author(s)

Laura Marshall

Examples

```
# The default values create a detectability object with a half normal
# detection function with scale parameter 25 and truncation distance 50.
detect <- make.detectability()
detect
```

```
# To include covariate parameters which affect detecability,
# first you need to make sure the population has covariates defined
# see examples in ?make.population.description
# Multi-strata covariate example
# Make a multi strata region
poly1 <- data.frame(x = c(0,0,100,100,0), y = c(0,100,100,0,0))
poly2 <- data.frame(x = c(200,200,300,300,200), y = c(10,110,110,10,10))
coords <- list(list(poly1), list(poly2))
region <- make.region(coords = coords)
density <- make.density(region)
# Create the population description
covariate.list <- list()
covariate.list$size <- list(list("ztruncpois", list(mean = 3)),
                            list("ztruncpois", list(mean = 5)))
covariate.list$height <- list(list("lognormal", list(meanlog = log(2), sdlog = log(1.25))))
covariate.list$sex <- list(data.frame(level = c("male", "female"), prob = c(0.45,0.55)),
                           data.frame(level = c("male", "female"), prob = c(0.5,0.5)))
pop.desc <- make.population.description(region.obj = region,
                                        density.obj = density,
                                        covariates = covariate.list,
                                        N = c(10,10))

# In this example height and sex have a global effect where as the effects of size on
# detectability vary by strata.
cov.params <- list(size = c(log(1.05), log(1.1)),
                   height = log(1.2),
                   sex = data.frame(level = c("male", "female"),
                                    param = c(log(1), log(0.6))))

detect <- make.detectability(key.function = "hn", scale.param = 20,
                             truncation = 50, cov.param = cov.params)

plot(detect, pop.desc)

# If we want the effects of sex to be strata specific we can define detectability as follows:
cov.params <- list(size = c(0.05, 0.1),
                   height = 0.2,
                   sex = data.frame(level = c("male", "female","male", "female"),
                                    strata = c("A", "A", "B", "B"),
                                    param = c(0,-0.5, 0.1, -0.25)))

detect <- make.detectability(key.function = "hn", scale.param = c(20, 25),
                             truncation = 60, cov.param = cov.params)
plot(detect, pop.desc)
```

---

make.population.description

Creates a Population.Description object

Description

Creates an object which describes a population. The values in this object will be used to create instances of the population

Usage

```
make.population.description(region.obj = make.region(),
  density.obj = make.density(), covariates = list(), N = numeric(0),
  fixed.N = TRUE)
```

Arguments

region.obj	the Region object in which this population exists (see make.region).
density.obj	the Density object describing the distribution of the individuals / clusters (see make.density).
covariates	Named list with one named entry per individual level covariate. Cluster sizes can be defined here. Each list entry should be another list with either one element or one element per strata allowing different population structures per strata. Each element of these lists should either be a data.frame containing 2 columns, the first the level (level) and the second the probability (prob). The cluster size entry in the list must be named 'size'. Alternatively the list element may be another list specifying the distribution in the first element and a named list in the second element with the distribution parameter.
N	the number of individuals / clusters in a population (1000 by default)
fixed.N	a logical value. If TRUE the population is generated from the value of N otherwise it is generated from the density description.

Details

#' The covariates argument should specify a list with one named element per covariate. If specifying the covariate values via a distribution this should be done in the form of a list. The first element should be one of the following: 'normal', 'poisson', 'ztruncpois' or 'lognormal'. The 'ztruncpois' distribution refers to a zero truncated Poisson distribution. The corresponding parameters that you must supply are detailed below. These should be added to a named list (each element named with the parameter name) containing the parameter values. See examples for implementation.

Distribution	Parameters	
normal	mean	sd
poisson	lambda	
ztruncpois	mean	
lognormal	meanlog	sdlog

Value

object of class Population.Description

Author(s)

Laura Marshall

See Also

make.region, make.density, make.detectability

Examples

```
# An example population can be created from the default values:
# - the default region
# - a constant density surface
# - and a population size of 1000
pop.desc <- make.population.description()

# To view an instance of this population
pop <- generate.population(pop.desc, make.detectability(), make.region())
plot(make.region())
plot(pop)

# An example population with covariates which vary by strata
# Make a multi strata region
poly1 <- data.frame(x = c(0,0,100,100,0), y = c(0,100,100,0,0))
poly2 <- data.frame(x = c(200,200,300,300,200), y = c(10,110,110,10,10))
coords <- list(list(poly1), list(poly2))
region <- make.region(coords = coords)
density <- make.density(region)

# Cluzter size is a zero truncated poisson with mean = 5 in strata 1 and a poisson with
# lambda = 30 in strata 2.
covariate.list <- list()
covariate.list$size <- list(list("ztruncpois", list(mean = 5)),
                            list("poisson", list(lambda = 30)))

# Animal height is generated from a lognormal distribution for both strata
covariate.list$height <- list(list("lognormal", list(meanlog = log(2), sdlog = log(1.25))))

# Animal sex is discrete/categorical, there are more females than males in strata 1 and equal
# numbers in strata 2
covariate.list$sex <- list(data.frame(level = c("male", "female"), prob = c(0.45,0.55)),
                           data.frame(level = c("male", "female"), prob = c(0.5,0.5)))

# Create covariate description
pop.desc <- make.population.description(region.obj = region,
                                        density.obj = density,
                                        covariates = covariate.list,
                                        N = c(10,10))

# To view the covariate values
pop <- generate.population(pop.desc, detect = make.detectability(), region)
pop@population
# Note that the covariate values have not affected the detectability (the scale parameter) to
```

```
# do this we need to set the cov.param argument in make.detectability. See ?make.detectability
```

`make.region` *Creates a Region object*

Description

This creates an instance of the Region class. If the `shapefile` argument is supplied, all information will be extracted from there. Otherwise, the a list of polygons describing the areas of interest needs to be supplied (`coords`) and optionally a list of polygons describing the areas to be excluded (`gaps`). If `area` is not specified it will be calculated.

Usage

```
make.region(region.name = "region", strata.name = character(0),
  units = "m", area = numeric(0), shapefile = NULL, coords = coords
  <- list(list(data.frame(x = c(0, 0, 2000, 2000, 0), y = c(0, 500, 500, 0,
  0)))), gaps = list(), check.LinkID = TRUE)
```

Arguments

`region.name`	the region name
`strata.name`	the stratum names (character vector, same length as the number of areas in the `shapefile` or `coords` arguments). If not supplied "A", "B", "C", ... will be assigned.
`units`	measurement units; either `"m"` for metres or `"km"` for kilometres.
`area`	the area of the region (optional - if not supplied it will be calculated for you)
`shapefile`	a shapefile object of the region loaded into R using `read.shapefile(shape.name)` from the shapefiles library.
`coords`	A list with one element per stratum. Each element in the list is a list of dataframes describing the polygon coordinates. This allows multiple regions in each strata. The coordinates should start and finish with the same point. By default DSsim will create a rectangular study region 2000 m by 500 m.
`gaps`	A list with one element per stratum giving the areas to be excluded from the study area (the "holes"). Each element in the list is a list of `data.frames` describing the polygon coordinates. This allows multiple gaps in each stratum. The corrdinates should start and finish with the same point.
`check.LinkID`	boolean to check the order of the LinkID value in the attribute table. This is important if this shapefile was used in Distance to create the survey shapefiles as Distance would have re-ordered the strata in this way. Failing to re-order the strata will mean that the strata in DSsim will not match the transect strata ID values created by Distance. If you have created your surveys outside Distance you can turn this option off.

Value

object of class Region

Author(s)

Laura Marshall

Examples

```
# A basic study region of 2000m by 500m is created using the defaults
region <- make.region()
plot(region)

# Here is an example of a 1000 x 1000 study region with a gap
coords <- gaps <- list()
coords[[1]] <- list(data.frame(x = c(0,1000,1000,0,0), y = c(0,0,
 1000,1000,0)))
gaps[[1]] <- list(data.frame(x = c(400,600,500,350,400), y = c(100,
 250,600,120,100)))

region <- make.region(region.name = "study.area", units = "m",
 coords = coords, gaps = gaps)
plot(region)
```

---

make.simulation *Creates a Simulation object*

Description

This creates a simulation with all the information necessary for DSsim to generate a population, create or read in transects, simulate the survey process and fit detection functions and estimate density / abundance. This function can be used by itself based on default values to create a simple line transect example, see Examples below. To create more complex simulations it is advisable to define the different parts of the simulation individually before grouping them together. See the Arguments for links to the functions which make the definitions for the individual simulation components. Example simulations can also be found at <https://github.com/DistanceDevelopment/DSsim/wiki>.

Usage

```
make.simulation(reps = 10, single.transect.set = FALSE,
  double.observer = FALSE, region.obj = make.region(),
  design.obj = make.design(),
  population.description.obj = make.population.description(),
  detectability.obj = make.detectability(),
  ddf.analyses.list = make.ddf.analysis.list())
```

Arguments

`reps`	number of times the simulation should be repeated
`single.transect.set`	logical specifying whether the transects should be kept the same throughout the simulation.
`double.observer`	not currently implemented.
`region.obj`	an object of class Region created by a call to make.region
`design.obj`	an object of class Survey.Design created by a call to make.design
`population.description.obj`	an object of class Population.Description created by a call to make.population.description
`detectability.obj`	and object of class Detectabolity created by a call to make.detectability
`ddf.analyses.list`	a list of objects of class DDF.Analysis created by a call tomake.ddf.analysis.list

Details

The `make.simulation` function is now set up so that by default (with the exception of specifying point transects rather than line) it can run a simple simulation example. See examples.

Value

object of class Simulation

Author(s)

Laura Marshall

Examples

```
## Not run:
# A basic line transect simulation example
sim <- make.simulation()
check.sim.setup(sim)
sim <- run(sim)
summary(sim)

# A basic point transect simulation example
sim <- make.simulation(design.obj = make.design("point"))
check.sim.setup(sim)
sim <- run(sim)
summary(sim)
# Note % bias levels will vary due to low number of repetitions
# set by default in these examples

# To increase the number of repetitions
sim <- make.simulation(reps = 100)
sim <- run(sim)
```

```
summary(sim)

## End(Not run)

coords <- gaps <- list()
coords[[1]] <- list(data.frame(x = c(0,1000,1000,0,0), y = c(0,0,
 1000,1000,0)))
gaps[[1]] <- list(data.frame(x = c(400,600,500,350,400), y = c(100,
 250,600,120,100)))

region <- make.region(region.name = "study.area", units = "m",
 coords = coords, gaps = gaps)
plot(region)

## Not run:
data(transects.shp)
#Edit the pathway below to point to an empty folder where the
#transect shapefile will be saved
shapefile.pathway <- "C:/..."
write.shapefile(transects.shp, paste(shapefile.pathway,"/transects_1",
 sep = ""))

parallel.design <- make.design(transect.type = "Line",
 design.details = c("Parallel","Systematic"), region = region,
 design.axis = 0, spacing = 100, plus.sampling =FALSE,
 path = shapefile.pathway)

pop.density <- make.density(region.obj = region, x.space = 10,
 y.space = 10, constant = 0.5)
pop.density <- add.hotspot(pop.density, centre = c(50, 200),
 sigma = 100, amplitude = 0.1)
pop.density <- add.hotspot(pop.density, centre = c(500, 700),
 sigma = 900, amplitude = 0.05)
pop.density <- add.hotspot(pop.density, centre = c(300, 100),
 sigma = 100, amplitude = -0.15)

plot(pop.density)
plot(region, add = TRUE)

pop.description <- make.population.description(N = 1000,
 density.obj = pop.density, region = region, fixed.N = TRUE)

detect <- make.detectability(key.function = "hn", scale.param = 15,
 truncation = 30)

ddf.analyses <- make.ddf.analysis.list(dsmodel = list(~cds(key = "hn",
 formula = ~1),~cds(key = "hr", formula = ~1)), method = "ds",
 criteria = "AIC")

my.simulation <- make.simulation(reps = 10, single.transect.set = TRUE,
 region.obj = region, design.obj = parallel.design,
 population.description.obj = pop.description,
 detectability.obj = detect, ddf.analyses.list = ddf.analyses)
```

```
survey.results <- create.survey.results(my.simulation, dht.table = TRUE)

plot(survey.results)

my.simulation <- run(my.simulation)

summary(my.simulation)

## End(Not run)
```

`Obs.Table-class` *Class "Obs.Table"*

Description

Class "`Obs.Table`" is an S4 class containing an observation table which is required for Hortvitz-Thompson estimation of density and abundance.

Slots

`obs.table` data.frame for dht

Objects from the Class

Objects can be created by calls to the function `create.survey.results(simulation, dht.table = TRUE)`

See Also

`create.survey.results`

`param.list` *Parameter Estimates from Covariate Simulation*

Description

This is a list of 2 2D arrays. Each array contains parameter estimates from the covariate simulations described in the vignette. These simulations generate data based on sex affecting detectability, they then fit a model to these data with sex as a covariate to see if the original parameters can be estimated from the data. This process was repeated 999 times for 5 different truncation values to see if truncation distance affects the parameter estimates.

Usage

```
data("cov_param")
```

Format

The format is: List of 2 $sigma, $sex.male ...

`sigma` a numeric 2D array

`sex.male` a numeric 2D array

Examples

```
data(cov_param)
```

---

plot,DDF.Data,ANY-method
Plot

Description

Plots an S4 object of class 'DDF.Data'. Requires that the associated region has already been plotted. This function adds the locations of the individuals/clusters in the population who were detected.

Usage

```
## S4 method for signature 'DDF.Data,ANY'
plot(x, y, ...)
```

Arguments

`x`	object of class DDF.Data
`y`	not used
`...`	other general plot parameters

---

plot,Density,ANY-method
Plot

Description

Plots an S4 object of class 'Density'

Usage

```
## S4 method for signature 'Density,ANY'
plot(x, y, add = FALSE,
  plot.units = character(0), contours = TRUE, style = "points",
  density.col = heat.colors(12), main = "", ...)
```

Arguments

x	object of class Density
y	not used
add	logical indicating whether it should be added to existing plot
plot.units	allows for units to be converted between m and km
contours	logical indicating whether contours should be added
style	character "points" or "blocks". Points displays a coloured point at the centre of each grid cell where as blocks colours the entire cell.
density.col	the colours used to indicate density level
main	character plot title
...	other general plot parameters

---

plot,Detectability,ANY-method
Plot

Description

Plots an S4 object of class 'Detectability'

Usage

```
## S4 method for signature 'Detectability,ANY'
plot(x, y, add = FALSE,
  plot.units = character(0), region.col = NULL, gap.col = NULL,
  main = "", ...)

## S4 method for signature 'Detectability,Population.Description'
plot(x, y, add = FALSE,
  plot.units = character(0), region.col = NULL, gap.col = NULL,
  main = "", ...)
```

Arguments

x	object of class Detectability
y	object of class Population.Description
add	logical indicating whether it should be added to existing plot
plot.units	allows for units to be converted between m and km
region.col	fill colour for the region
gap.col	fill colour for the gaps
main	character plot title
...	other general plot parameters

plot,Line.Transect,ANY-method
Plot

Description

Plots an S4 object of class 'Line.Transect'. Requires that the associated region has already been plotted. This function adds the transect lines.

Usage

```
## S4 method for signature 'Line.Transect,ANY'
plot(x, y, transect.ID = numeric(0),
  col = 1, ...)
```

Arguments

x	object of class Line.Transect
y	not used
transect.ID	allows individual or groups of transects to be added
col	colour of the lines
...	other general plot parameters e.g. lwd

plot,LT.Survey.Results,ANY-method
Plot

Description

Plots an S4 object of class 'LT.Survey.Results'. Plots the region, the transects, the population and colour codes the detections

Usage

```
## S4 method for signature 'LT.Survey.Results,ANY'
plot(x, y, ...)
```

Arguments

x	object of class LT.Survey.Results
y	not used
...	other general plot parameters

plot,Point.Transect,ANY-method
Plot

Description

Plots an S4 object of class 'Point.Transect'. Requires that the associated region has already been plotted. This function adds the transect lines.

Usage

```
## S4 method for signature 'Point.Transect,ANY'
plot(x, y, transect.ID = numeric(0),
  col = 1, ...)
```

Arguments

x	object of class Point.Transect
y	not used
transect.ID	allows individual or groups of transects to be added
col	colour of the lines
...	other general plot parameters e.g. lwd

plot,Population,ANY-method
Plot

Description

Plots an S4 object of class 'Population'. Requires that the associated region has already been plotted. This function adds the locations of the individuals/clusters in the population.

Usage

```
## S4 method for signature 'Population,ANY'
plot(x, y, ...)
```

Arguments

x	object of class Population
y	not used
...	other general plot parameters

plot,Region,ANY-method *Plot*

Description

Plots an S4 object of class 'Region'

Usage

```
## S4 method for signature 'Region,ANY'
plot(x, y, add = FALSE,
  plot.units = character(0), region.col = NULL, gap.col = NULL,
  main = "", ...)
```

Arguments

x	object of class Region
y	not used
add	logical indicating whether it should be added to existing plot
plot.units	allows for units to be converted between m and km
region.col	fill colour for the region
gap.col	fill colour for the gaps
main	character plot title
...	other general plot parameters

plot,Survey.Results,ANY-method *Plot*

Description

Plots an S4 object of class 'Survey.Results'. Plots the region, the transects, the population and colour codes the detections

Usage

```
## S4 method for signature 'Survey.Results,ANY'
plot(x, y, ...)
```

Arguments

x	object of class LT.Survey.Results
y	not used
...	other general plot parameters

Point.Transect-class *S4 Class "Point.Transect"*

Description

Class "Point.Transect" contains an instance of a Line Transect Survey

Slots

`design.obj` Object of class `"character"`; the object name of the design object which generated the transects.

`sampler.info` Object of class `"data.frame"`; the sampler point coordinates.

Methods

`plot signature=(object = "Point.Transect")`: plots the transects.

Population-class *Class "Population"*

Description

Contains an instance of a population including a description of their detectability in the form of an object of class Detectability.

Slots

`region.obj` Object of class `"character"`; the name of the region object.

`strata.names` Object of class `"character"`; the names of the strata.

`N` Object of class `"numeric"`; the number of individuals/clusters.

`D` Object of class `"numeric"`; the density of individuals/clusters.

`population` Object of class `"data.frame"`; the locations of individuals/clusters and any population covariates.

`detectability` Object of class `"Detectability"`; describes how easily the individuals/clusters can be detected.

Methods

`plot signature=(object = "Line.Transect")`: plots the locations of the individuals/clusters.

See Also

`make.population.description`, `make.detectability`

Population.Description-class
Class "Population.Description"

Description

Class "Population.Description" is an S4 class containing a description of the population. It provides methods to generate an example population.

Slots

N Object of class "numeric"; number of individuals in the population (optional).

density Object of class "Density"; describes the population density

region.name Object of class "character"; name of the region in which the population exists.

strata.names Character vector giving the strata names for the study region.

covariates Named list with one named entry per individual level covariate. Cluster sizes can be defined here. Each list entry will either be a data.frame containing 2 columns, the first the level (level) and the second the probability

size logical value indicating whether the population occurs in clusters. (prob). The cluster size entry in the list must be named 'size'.

gen.by.N Object of class "logical"; If TRUE N is fixed otherwise it is generated from a Poisson distribution.

D.dist Object of class character; Describes the density distribution (currently not implemented).

Methods

get.N signature=(object = "Population.Description"): returns the value of N

generate.population signature=(object = "Population.Description"): generates a single realisation of the population.

See Also

make.population.description

Population.Summary-class

S4 Class "Population.Summary"

Description

Class "Population.Summary"

Details

Class "Population.Summary" is an S4 class containing a summary of the survey population. This is returned when summary(Population) is called. If it is not assigned to a variable the object will be displayed via the show method.

Methods

show signature=(object = "Population.Summary"): prints the contents of the object in a user friendly format.

PT.Design-class

Virtual Class "PT.Design" extends Class "Survey.Design"

Description

Virtual Class "PT.Design" is an S4 class detailing the type of line transect design.

Methods

generate.transects signature=(object = "PT.Design", ...): loads a set of transects from a shapefile.

See Also

make.design

PT.Survey-class *Virtual Class "PT.Survey" extends class "Survey"*

Description

Virtual Class "PT.Survey" is an S4 class containing a population and a set of transects.

Slots

transect Object of class "Transect"; the transects.

radial.truncation Object of class "numeric"; the maximum distance from the transect at which animals may be detected.

Methods

create.sample.table signature=(object = "PT.Survey", ...): creates a sample table for dht.

See Also

make.design

Region-class *Class "Region"*

Description

Class "Region" is an S4 class containing descriptions of the study area. The polygons describing the region are found in the coords slot and any gaps are described as polygons in the gaps slot.

Slots

region.name Object of class "character"; giving the name of the region.

strata.name Object of class "character"; character vector giving the names of the strata.

strata.name Object of class "character"; character vector giving the names of the strata.

units Object of class "character"; character describing the coordinate units ("km" or "m")

area Object of class "numeric"; the area of the survey region

box Object of class "numeric"; 4 values giving the x and y ranges of the region

coords Object of class "list"; this list contains an element for each strata. Each of these list elements contains a list of polygons defining the region.

gaps Object of class "list";this list contains an element for each strata. Each of these list elements contains a list of gaps in the region

Objects from the Class

Objects can be created by calls of the form `make.region(region.name = "region.name", shapefile = region.shapefi`

Methods

`get.area signature(obj = "Region")`: retrieves the area element

`plot signature(x = "Region", y = "missing")`: plots the survey region defined by the object.

See Also

`make.region`

Region.Table-class *S4 Class "Region.Table"*

Description

Class "Region.Table"

Details

Class `"Region.Table"` is an S4 class containing a region table which is required for Hortvitz-Thompson estimation of density and abundance.

Slots

`region.table` data.frame which is a region.table for dht

Objects from the Class

Objects can be created by calls to the function `create.survey.results(simulation, dht.table = TRUE)`

See Also

`create.survey.results`

`rename.duplicates` *Renumbers the object IDs for the duplicate observations generated when bootstrapping*

Description

Find the largest object ID and renumbers all duplicate IDs starting form this value. The information for the duplicates is also added to the obs.table

Usage

```
rename.duplicates(ddf.dat, double.observer = FALSE)
```

Arguments

`ddf.dat`	dataframe containing a single dataset with duplicate observations
`double.observer`	logical indicating whether it is a double observer survey or not

Value

list with 2 elements: ddf.dat dataframe containing a single dataset with new and unique observation IDs obs.table the updated obs.table dataframe containing the new observation IDs

Note

Internal function not intended to be called by user.

Author(s)

Laura Marshall

See Also

`resample.data`

rtpois *Randomly generates values from a zero-truncated Poisson distribution*

Description

Generates values from a zero-truncated Poisson distribution with mean equal to that specified. It uses a look up table to check which value of lambda will give values with the requested mean.

Usage

```
rtpois(N, mean = NA)
```

Arguments

N	number of values to randomly generate
mean	mean of the generated values

Note

Internal function not intended to be called by user.

Author(s)

Laura Marshall

run *S4 generic method to run a simulation*

Description

Runs the simulation and returns the simulation object with results. If running in parallel and max.cores is not specified it will default to using one less than the number of cores / threads on your machine.

Usage

```
run(object, run.parallel = FALSE, max.cores = NA, ...)

## S4 method for signature 'Simulation'
run(object, run.parallel = FALSE,
  max.cores = NA, save.data = FALSE, load.data = FALSE,
  data.path = character(0), counter = TRUE, progress.file = "")
```

Arguments

object	an object of class Simulation
run.parallel	logical option to use multiple processors
max.cores	integer maximum number of cores to use, if not specified then one less than the number available will be used.
...	allows the five previous optional arguments to be specified
save.data	logical allows the datasets from the simulation to be saved to file
load.data	logical allows the datasets to be loaded from file rather than simulated afresh.
data.path	character file path to the data files.
counter	logical can be used to turn off simulation counter when running in serial.
progress.file	character file to output progress to for Distance for Windows

Value

an object of class simulation which now includes the results

See Also

make.simulation

---

run.analysis *S4 generic method to run analyses*

Description

This method carries out an analysis of distance sampling data. This method is provided to allow the user to perform diagnostics of the analyses used in the simulation. The data argument can be obtained by a call to `simulate.survey(object, dht.table = TRUE)`. Note if the first object supplied is of class DDf.Analysis then the second argument must be of class DDf.Data. The data argument may be of either class for an object argument of class Simulation.

Usage

```
run.analysis(object, data, ...)

## S4 method for signature 'DDF.Analysis,DDF.Data'
run.analysis(object, data, dht = FALSE,
  point = FALSE, warnings = list())

## S4 method for signature 'Simulation,Survey.Results'
run.analysis(object, data,
  dht = FALSE)

## S4 method for signature 'Simulation,DDF.Data'
run.analysis(object, data, dht = FALSE)
```

Arguments

object	an object of class Simulation or DDF.Analysis
data	an object of class Survey.Results or DDF.Data
...	optional arguments including the following:
dht	logical whether density should be estimated after fitting the model
point	logical indicating whether it is a point transect survey
warnings	a list of warnings and how many times they arose

Value

a list containing an S3 ddf object and optionally an S3 dht object relating to the model with the minimum criteria.

Sample.Table-class *Class "Sample.Table"*

Description

Class "Sample.Table" is an S4 class containing a region table which is required for Hortvitz-Thompson estimation of density and abundance.

Slots

sample.table data.frame which is the sample table for dht

Objects from the Class

Objects can be created by calls to the function create.survey.results(simulation, dht.table = TRUE)

See Also

create.survey.results

save.sim.results *save.sim.results*

Description

Saves the simulation results from each replicate to file. It will save up to 3 csv files, one for the abundance estimation for individuals, one for the abundance estimation of clusters (where applicable) and one for detectability estimates and model selection information.

Usage

```
save.sim.results(simulation, filepath = character(0),
  sim.ID = numeric(0))
```

Arguments

simulation	object of class Simulation which has been run.
filepath	optionally a path to the directory where you would like the files saved, otherwise it will save it to the working directory.
sim.ID	optionally you can add a simulation ID to the filename

Value

invisibly returns the original simulation object

Author(s)

L. Marshall

show,Design.Summary-method *show*

Description

Displays the simulation summary

Displays the simulation summary

Usage

```
## S4 method for signature 'Design.Summary'
show(object)

## S4 method for signature 'Simulation.Summary'
show(object)
```

Arguments

object	object of class Simulation.Summary

Details

#@param object object of class Simulation.Summary

---

show,Simulation-method *show*

Description

Not currently implemented

Usage

```
## S4 method for signature 'Simulation'
show(object)
```

Arguments

object	object of class Simulation

Simulation-class *Class "Simulation"*

Description

Class "Simulation" is an S4 class containing descriptions of the region, population, survey design and analyses the user wishes to investigate. Once the simulation has been run the N.D.Estimates will contain multiple estimates of abundance and density obtained by repeatedly generating populations, simulating the survey and completing the analyses.

Slots

reps Object of class "numeric"; the number of times the simulation should be repeated.

single.transect.set Object of class "logical"; if TRUE the same set of transects are used in each repetition.

double.observer Object of class "logical"; whether a double observer protocol is being used. Not currently implemented.

region Object of class "Region"; the survey region.

design Object of class "Survey.Design"; the survey design.

population.description Object of class "Population.Description"; the population.description.

`detectability` Object of class `"Detectability"`; a description of the detectability of the population.

`ddf.analyses` Object of class `"list"`; a list of objects of class DDF.Analysis. These are fitted and the one with the minimum criteria is selected and used in predicting N and D.

`dsm.analysis` Object of class `"DSM.Analysis"`; Not yet implemented.

`ddf.param.ests` Object of class `"array"`; stores the parameters associated with the detection function.

`results` A `"list"` of `"arrays"`; stores the estimated of abundance and density as well as other measures of interest.

`warnings` A `"list"` to store warnings and error messages encountered during runtime.

Methods

`add.hotspot signature=(object = "Simulation")`: adds a hotspot based on a gaussian decay to the density surface.

`summary signature=(object = "Simulation")`: produces a summary of the simulation and its results.

`generate.population signature = (object = "Simulation")`: generates a single instance of a population.

`generate.transects signature = (object = "Simulation")`: generates a single set of transects.

`create.survey.results signature = (object = "Simulation")`: carries out the simulation process as far as generating the distance data and returns an object containing the population, transects and data.

`run.analysis signature = c(object = "Simulation", data = "LT.Survey.Results")`: returns the ddf analysis results from the models in the simulation fitted to the data in the LT.Survey.Results object.

`run.analysis signature = c(object = "Simulation", data = "DDF.Data")`: returns the ddf analysis results from the models in the simulation fitted to the data in the DDF.Data object.

`run signature = (object = "Simulation")`: runs the whole simulation for the specified number of repetitions.

See Also

`make.simulation`

Simulation.Summary-class

Class "Simulation.Summary"

Description

Class `"Simulation.Summary"` is an S4 class containing a summary of the simulation results. This is returned when `summary(Simulation)` is called. If it is not assigned to a variable the object will be displayed via the `show` method.

Methods

`show signature=(object = "Simulation.Summary")`: prints the contents of the object in a user friendly format.

`Single.Obs.DDF.Data-class`
S4 Class "Single.Obs.DDF.Data"

Description

DDF data resulting from a single observer survey.

`Single.Obs.LT.Survey-class`
Class "Single.Obs.LT.Survey"

Description

An S4 class containing an instance of a population and a set of transects.

`Single.Obs.PT.Survey-class`
Class "Single.Obs.PT.Survey"

Description

An S4 class containing an instance of a population and a set of transects.

summary,Simulation-method *summary*

Description

Provides a summary of the simulation results.

Usage

```
## S4 method for signature 'Simulation'
summary(object, description.summary = TRUE, ...)
```

Arguments

object	object of class Simulation
description.summary	logical indicating whether an explanation of the summary should be included
...	can specify if you want the maximum number of iterations to be used where at least one model converged (use.max.reps = TRUE) or only use iterations where all models converged (use.max.reps = FALSE)

summary.list *Truncation Simulation Summaries*

Description

This is a list of 3 simulation summaries. Data were generated using a systematic line transect design and a single half normal detection function describing detectability. Each of the three simulation summaries corresponds to a different analysis truncation distance, to test if this affects the accuracy and precision of the estimates of abundance / density.

Usage

```
data("trunc_summary")
```

Format

The format is: List of 5

`t200` a simulation summary (truncation = 200)

`t400` a simulation summary (truncation = 400)

`t600` a simulation summary (truncation = 600)

Examples

```
data(trunc_summary)
summary.list$t200
```

`Survey-class` *Virtual Class "Survey"*

Description

Class "Survey" is an S4 class containing an instance of a population.

Slots

`population` Object of class "Population"; an instance of a population.

`transect` Object of class "Transect"; the transects.

Methods

`create.region.table signature=(object = "Survey", ...)`: creates a region table for dht.

`create.sample.table signature=(object = "Survey", ...)`: creates a sample table for dht.

`Survey.Design-class` *S4 Class "Survey.Design"*

Description

Virtual Class "Survey.Design"

Details

Virtual Class "Survey.Design" is an S4 class detailing the survey design. Currently only line transect designs are implemented and transects from these designs must be generated using the Distance software in advance.

Slots

`design.axis` Object of class "numeric"; the angle of the design axis.

`spacing` Object of class "numeric"; the spacing of the design.

`region.obj` Object of class "character"; The name of the region which the survey design has been made for.

`plus.sampling` Object of class "logical"; Whether a plus sampling protocol is to be used.

`path` Object of class "character"; Describing the folder where the shapefiles containing the transects are located.

`filenames` Object of class "character"; stores the filenames of the transect shapefiles. These are automatically added when the object is created using all the files in the specified path.

`file.index` Object of class "numeric"; Keeps track of which shapefile is to be loaded.

See Also

make.design

Survey.Results-class *S4 Class "Survey.Results"*

Description

Class containing all the components relating to a single realisation of a survey.

Slots

region Object of class "Region"; the region representation.

population Object of class "Population"; the population.

transects Object of class "Transect"; the transects.

ddf.data Object of class "Single.Obs.DDF.Data"; The ddf data for ddf. @slot obs.table Object of class "Obs.Table"; One of the tables for dht. @slot sample.table Object of class "Sample.Table"; One of the tables for dht. @slot region.table Object of class "Region.Table"; One of the tables for dht.

Methods

plot signature=(object = "Survey.Results"): plots the region, the location of individuals in the population, the transects and the successful sightings.

get.distance.data signature=(object = "Survey.Results"): returns the ddf data as a dataframe..

Transect-class *S4 Virtual Class "Transect"*

Description

Class "Transect" contains an instance of a Line Transect Survey

Slots

design.obj Object of class "character"; the object name of the design object which generated the transects.

sampler.info Object of class "data.frame"; the sampler coordinates.

Methods

plot signature=(object = "Transect"): plots the transects.

`transects.shp` *Shapefile describing the transects*

Description

Shapefile object of type polyline which describes one set of transects in the study region example.

Format

The format is: List of 3 $ shp, $shx, $dbf ...

Index