Package 'Distance'

May 1, 2019

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

License GPL (>= 2)

Title Distance Sampling Detection Function and Abundance Estimation

LazyLoad yes

Author David Lawrence Miller

Description A simple way of fitting detection functions to distance sampling data for both line and point transects. Adjustment term selection, left and right truncation as well as monotonicity constraints and binning are supported. Abundance and density estimates can also be calculated (via a Horvitz-Thompson-like estimator) if survey area information is provided.

Version 0.9.8

URL http://github.com/DistanceDevelopment/Distance/

BugReports https://github.com/DistanceDevelopment/Distance/issues

Depends R (>= 3.0), mrds (>= 2.2.0)

Suggests testthat

RoxygenNote 6.1.1

NeedsCompilation no

Repository CRAN

Date/Publication 2019-05-01 16:50:03 UTC

R topics documented:

`Distance-package` *Distance sampling*

Description

`Distance` is a simple way to fit detection functions and estimate abundance using distance sampling methodology.

Details

Underlying `Distance` is the package `mrds`, for more advanced analyses (such as those involving double observer surveys) one may find it necessary to use `mrds`.

Further information on distance sampling methods and example code is available at `http://distancesampling.org/R/`.

For help with distance sampling and this package, there is a Google Group `https://groups.google.com/forum/#!forum/distance-sampling`.

Author(s)

David L. Miller <dave@ninepointeightone.net>

References

Key References:

Miller D.L., E. Rexstad, L. Thomas, L. Marshall and J.L. Laake. 2019. Distance Sampling in R. Journal of Statistical Software, 89(1), 1-28. `http://doi.org/10.18637/jss.v089.i01`.

Background References:

Laake, J.L. and D.L. Borchers. 2004. Methods for incomplete detection at distance zero. In: Advanced Distance Sampling, eds. S.T. Buckland, D.R.Anderson, K.P. Burnham, J.L. Laake, D.L. Borchers, and L. Thomas. Oxford University Press.

Marques, F.F.C. and S.T. Buckland. 2004. Covariate models for the detection function. In: Advanced Distance Sampling, eds. S.T. Buckland, D.R.Anderson, K.P. Burnham, J.L. Laake, D.L. Borchers, and L. Thomas. Oxford University Press.

`AIC.dsmodel` *Akaike's An Information Criterion for detection functions*

Description

Extract the AIC from a fitted detection function.

Usage

```
## S3 method for class 'dsmodel'
AIC(object, ..., k = 2)
```

Arguments

`object`	a fitted detection function object
`...`	optionally more fitted model objects.
`k`	penalty per parameter to be used; the default `k = 2` is the "classical" AIC

Author(s)

David L Miller

Examples

```
## Not run:
library(Distance)
data(minke)
model <- ds(minke, truncation=4)
model_hr <- ds(minke, truncation=4, key="hr")
# extract the AIC for 2 models
AIC(model, model_hr)

## End(Not run)
```

`amakihi` *Amakihi (Hemignathus virens) point transect data*

Description

1485 observations of Hawaiian amakihi. Data collected on Hawaii from point transects, collected as part of a larger study to assess a Palila (Loxioides bailleuli) translocation experiment on the island of Hawaii (Fancy et al. 1997) between July 1992 and April 1995 (7 survey periods). Full analyses of the amakihi data is provided in Marques et al (2007).

Format

1485 observations of 7 variables: `survey`, `object`, `distance`, `obs`, `mas`, `has`, `detected`.

Details

Data include distances, as well as survey period (`survey`, factor), observer code (`obs`, factor), hours after sunrise (`has`, factor) and minutes after sunrise (`mas`).

We thank Steve Fancy for making this data publicly available.

References

Fancy, SG, TJ Snetsinger, & JD Jacobi (1997). Translocation of the Palila, an Endangered Hawaiian Honeycreeper. Pacific Conservation, 3(1).

Marques, TA, L Thomas, S Fancy & ST Buckland (2007). Improving estimates of bird density using multiple-covariate distance sampling. The Auk, 124(4), 1229-1243.

`checkdata` *Check that the data supplied to* `ds` *is correct*

Description

This is an internal function that checks the `data.frames` supplied to `ds` are "correct".

Usage

```
checkdata(data, region.table = NULL, sample.table = NULL,
  obs.table = NULL, formula = ~1)
```

Arguments

`data`	as in `ds`
`region.table`	as in `ds`
`sample.table`	as in `ds`
`obs.table`	as in `ds`
`formula`	formula for the covariates

Value

Throws an error if something goes wrong, otherwise returns a `data.frame`.

Author(s)

David L. Miller

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

Description

This is an internal 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

Examples

```
## Not run:
library(Distance)
data(minke)

# put the minke data into bins 0-1, 1-2, 2-3 km
minke_cuts <- create.bins(minke[!is.na(minke$distance),], c(0,1,2,3))

## End(Not run)
```

`ds` *Fit detection functions and calculate abundance from line or point transect data*

Description

This function fits detection functions to line or point transect data and then (provided that survey information is supplied) calculates abundance and density estimates. The examples below illustrate some basic types of analysis using `ds()`.

Usage

```
ds(data, truncation = ifelse(is.null(cutpoints),
  ifelse(is.null(data$distend), max(data$distance), max(data$distend)),
  max(cutpoints)), transect = c("line", "point"), formula = ~1,
  key = c("hn", "hr", "unif"), adjustment = c("cos", "herm", "poly"),
  order = NULL, scale = c("width", "scale"), cutpoints = NULL,
  dht.group = FALSE, monotonicity = ifelse(formula == ~1, "strict",
  "none"), region.table = NULL, sample.table = NULL,
  obs.table = NULL, convert.units = 1, method = "nlminb",
  quiet = FALSE, debug.level = 0, initial.values = NULL,
  max.adjustments = 5)
```

Arguments

`data`	a `data.frame` containing at least a column called `distance` or a numeric vector containing the distances. NOTE! If there is a column called `size` in the data then it will be interpreted as group/cluster size, see the section "Clusters/groups", below. One can supply data as a "flat file" and not supply `region.table`, `sample.table` and `obs.table`, see "Data format", below and `flatfile`.
`truncation`	either truncation distance (numeric, e.g. 5) or percentage (as a string, e.g. "15%"). Can be supplied as a `list` with elements `left` and `right` if left truncation is required (e.g. `list(left=1,right=20)` or `list(left="1%",right="15%")` or even `list(left="1",right="15%")`). By default for exact distances the maximum observed distance is used as the right truncation. When the data is binned, the right truncation is the largest bin end point. Default left truncation is set to zero.
`transect`	indicates transect type "line" (default) or "point".
`formula`	formula for the scale parameter. For a CDS analysis leave this as its default `~1`.
`key`	key function to use; "hn" gives half-normal (default), "hr" gives hazard-rate and "unif" gives uniform. Note that if uniform key is used, covariates cannot be included in the model.
`adjustment`	adjustment terms to use; `"cos"` gives cosine (default), `"herm"` gives Hermite polynomial and `"poly"` gives simple polynomial. `"cos"` is recommended. A value of `NULL` indicates that no adjustments are to be fitted.
`order`	orders of the adjustment terms to fit (as a vector/scalar), the default value (`NULL`) will select via AIC up to `max.adjustments` adjustments. If a single number is given, that number is expanded to be `seq(term_min, order, by=1)` where `term.min` is the appropriate minimum order for this type of adjustment. For cosine adjustments, valid orders are integers greater than 2 (except when a uniform key is used, when the minimum order is 1). For Hermite polynomials, even integers equal or greater than 2 are allowed and for simple polynomials even integers equal or greater than 2 are allowed (though note these will be multiplied by 2, see Buckland et al, 2001 for details on their specification).
`scale`	the scale by which the distances in the adjustment terms are divided. Defaults to `"width"`, scaling by the truncation distance. If the key is uniform only `"width"` will be used. The other option is `"scale"`: the scale parameter of the detection

`cutpoints` — if the data are binned, this vector gives the cutpoints of the bins. Ensure that the first element is 0 (or the left truncation distance) and the last is the distance to the end of the furthest bin. (Default `NULL`, no binning.) Note that if `data` has columns `distbegin` and `distend` then these will be used as bins if `cutpoints` is not specified. If both are specified, `cutpoints` has precedence.

`dht.group` — should density abundance estimates consider all groups to be size 1 (abundance of groups) `dht.group=TRUE` or should the abundance of individuals (group size is taken into account), `dht.group=FALSE`. Default is `FALSE` (abundance of individuals is calculated).

`monotonicity` — should the detection function be constrained for monotonicity weakly (`"weak"`), strictly (`"strict"`) or not at all (`"none"` or `FALSE`). See Monotonicity, below. (Default `"strict"`). By default it is on for models without covariates in the detection function, off when covariates are present.

`region.table` — `data.frame` with two columns:

`Region.Label`	label for the region
`Area`	area of the region

`region.table` has one row for each stratum. If there is no stratification then `region.table` has one entry with `Area` corresponding to the total survey area.

`sample.table` — `data.frame` mapping the regions to the samples (i.e. transects). There are three columns:

`Sample.Label`	label for the sample
`Region.Label`	label for the region that the sample belongs to.
`Effort`	the effort expended in that sample (e.g. transect length).

`obs.table` — `data.frame` mapping the individual observations (objects) to regions and samples. There should be three columns:

`object`	unique numeric identifier for the observation
`Region.Label`	label for the region that the sample belongs to.
`Sample.Label`	label for the sample

`convert.units` — conversion between units for abundance estimation, see "Units", below. (Defaults to 1, implying all of the units are "correct" already.)

`method` — optimization method to use (any method usable by `optim` or **optimx**). Defaults to `"nlminb"`.

`quiet` — suppress non-essential messages (useful for bootstraps etc). Default value `FALSE`.

`debug.level` — print debugging output. `0`=none, `1-3` increasing levels of debugging output.

`initial.values` — a `list` of named starting values, see `mrds-opt`. Only allowed when AIC term selection is not used.

max.adjustments
: maximum number of adjustments to try (default 5) only used when `order=NULL`.

Value

a list with elements:

- `ddf` a detection function model object.
- `dht` abundance/density information (if survey region data was supplied, else `NULL`).

Details

If abundance estimates are required then the `data.frames` `region.table` and `sample.table` must be supplied. If `data` does not contain the columns `Region.Label` and `Sample.Label` then the `data.frame` `obs.table` must also be supplied. Note that stratification only applies to abundance estimates and not at the detection function level.

Clusters/groups

Note that if the data contains a column named `size` and `region.table`, `sample.table` and `obs.table` are supplied, cluster size will be estimated and density/abundance will be based on a clustered analysis of the data. Setting this column to be `NULL` will perform a non-clustered analysis (for example if "`size`" means something else in your dataset).

Truncation

The right truncation point is by default set to be largest observed distance or bin end point. This is a default will not be appropriate for all data and can often be the cause of model convergence failures. It is recommended that one plots a histogram of the observed distances prior to model fitting so as to get a feel for an appropriate truncation distance. (Similar arguments go for left truncation, if appropriate). Buckland et al (2001) provide guidelines on truncation.

When specified as a percentage, the largest `right` and smallest `left` percent distances are discarded. Percentages cannot be supplied when using binned data.

For left truncation, there are two options: (1) fit a detection function to the truncated data as is (this is what happens when you set `left`). This does not assume that g(x)=1 at the truncation point. (2) manually remove data with distances less than the left truncation distance – effectively move the centreline out to be the truncation distance (this needs to be done before calling `ds`). This then assumes that detection is certain at the left truncation distance. The former strategy has a weaker assumption, but will give higher variance as the detection function close to the line has no data to tell it where to fit – it will be relying on the data from after the left truncation point and the assumed shape of the detection function. The latter is most appropriate in the case of aerial surveys, where some area under the plane is not visible to the observers, but their probability of detection is certain at the smallest distance.

Binning

Note that binning is performed such that bin 1 is all distances greater or equal to cutpoint 1 (>=0 or left truncation distance) and less than cutpoint 2. Bin 2 is then distances greater or equal to cutpoint 2 and less than cutpoint 3 and so on.

Monotonicity

When adjustment terms are used, it is possible for the detection function to not always decrease with increasing distance. This is unrealistic and can lead to bias. To avoid this, the detection function can be constrained for monotonicity (and is by default for detection functions without covariates).

Monotonicity constraints are supported in a similar way to that described in Buckland et al (2001). 20 equally spaced points over the range of the detection function (left to right truncation) are evaluated at each round of the optimisation and the function is constrained to be either always less than it's value at zero ("weak") or such that each value is less than or equal to the previous point (monotonically decreasing; "strict"). See also check.mono in mrds.

Even with no monotonicity constraints, checks are still made that the detection function is monotonic, see check.mono.

Units

In extrapolating to the entire survey region it is important that the unit measurements be consistent or converted for consistency. A conversion factor can be specified with the convert.units variable. The values of Area in region.table, must be made consistent with the units for Effort in sample.table and the units of distance in the data.frame that was analyzed. It is easiest if the units of Area are the square of the units of Effort and then it is only necessary to convert the units of distance to the units of Effort. For example, if Effort was entered in kilometers and Area in square kilometers and distance in meters then using convert.units=0.001 would convert meters to kilometers, density would be expressed in square kilometers which would then be consistent with units for Area. However, they can all be in different units as long as the appropriate composite value for convert.units is chosen. Abundance for a survey region can be expressed as: A*N/a where A is Area for the survey region, N is the abundance in the covered (sampled) region, and a is the area of the sampled region and is in units of Effort * distance. The sampled region a is multiplied by convert.units, so it should be chosen such that the result is in the same units as Area. For example, if Effort was entered in kilometers, Area in hectares (100m x 100m) and distance in meters, then using convert.units=10 will convert a to units of hectares (100 to convert meters to 100 meters for distance and .1 to convert km to 100m units).

Data format

One can supply data only to simply fit a detection function. However, if abundance/density estimates are necessary further information is required. Either the region.table, sample.table and obs.table data.frames can be supplied or all data can be supplied as a "flat file" in the data argument. In this format each row in data has additional information that would ordinarily be in the other tables. This usually means that there are additional columns named: Sample.Label, Region.Label, Effort and Area for each observation. See flatfile for an example.

Author(s)

David L. Miller

References

Buckland, S.T., Anderson, D.R., Burnham, K.P., Laake, J.L., Borchers, D.L., and Thomas, L. (2001). Distance Sampling. Oxford University Press. Oxford, UK.

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

See Also

flatfile

Examples

```
# An example from mrds, the golf tee data.
library(Distance)
data(book.tee.data)
tee.data<-book.tee.data$book.tee.dataframe[book.tee.data$book.tee.dataframe$observer==1, ]
ds.model <- ds(tee.data, 4)
summary(ds.model)
plot(ds.model)

## Not run:
# same model, but calculating abundance
# need to supply the region, sample and observation tables
region <- book.tee.data$book.tee.region
samples <- book.tee.data$book.tee.samples
obs <- book.tee.data$book.tee.obs

ds.dht.model <- ds(tee.data, 4, region.table=region,
                   sample.table=samples, obs.table=obs)
summary(ds.dht.model)

# specify order 2 cosine adjustments
ds.model.cos2 <- ds(tee.data, 4, adjustment="cos", order=2)
summary(ds.model.cos2)

# specify order 2 and 3 cosine adjustments, turning monotonicity
# constraints off
ds.model.cos23 <- ds(tee.data, 4, adjustment="cos", order=c(2, 3),
                   monotonicity=FALSE)
# check for non-monotonicity -- actually no problems
check.mono(ds.model.cos23$ddf, plot=TRUE, n.pts=100)

# include both a covariate and adjustment terms in the model
ds.model.cos2.sex <- ds(tee.data, 4, adjustment="cos", order=2,
                        monotonicity=FALSE, formula=~as.factor(sex))
# check for non-monotonicity -- actually no problems
check.mono(ds.model.cos2.sex$ddf, plot=TRUE, n.pts=100)

# truncate the largest 10% of the data and fit only a hazard-rate
# detection function
ds.model.hr.trunc <- ds(tee.data, truncation="10%", key="hr",
                        adjustment=NULL)
summary(ds.model.hr.trunc)

# compare AICs between these models:
```

```
AIC(ds.model)
AIC(ds.model.cos2)
AIC(ds.model.cos23)

## End(Not run)
```

---

ds.gof *Goodness of fit tests for distance sampling models*

Description

Chi-square, Kolmogorov-Smirnov (if `ks=TRUE`) and Cramer-von Mises goodness of fit tests for detection function models.

Usage

```
ds.gof(model, breaks = NULL, nc = NULL, qq = TRUE, ks = FALSE, ...)
```

Arguments

`model`	fitted model object
`breaks`	Cutpoints to use for binning data
`nc`	Number of distance classes
`qq`	Flag to indicate whether quantile-quantile plot is desired
`ks`	perform the Kolmogorov-Smirnov test (this involves many bootstraps so can take a while)
`...`	Graphics parameters to pass into qqplot function

Value

List of test results and a plot.

Author(s)

David L Miller

See Also

qqplot.ddf ddf.gof

Examples

```
## Not run:
# fit and test a simple model for the golf tee data
library(Distance)
data(book.tee.data)
tee.data<-book.tee.data$book.tee.dataframe[book.tee.data$book.tee.dataframe$observer==1,]
ds.model <- ds(tee.data,4)
ds.gof(ds.model)

## End(Not run)
```

---

flatfile *The flatfile data format*

Description

`Distance` allows loading data as a "flat file" and analyse data (and obtain abundance estimates) straight away, provided that the format of the flat file is correct. One can provide the file as, for example, an Excel spreadsheet using `read.xls` in **gdata** or CSV using `read.csv`.

Details

Each row of the data table corresponds to one observation and must have a the following columns:

`distance`	observed distance to object
`Sample.Label`	Identifier for the sample (transect id)
`Effort`	effort for this transect (e.g. line transect length or number of times point transect was visited)
`Region.Label`	label for a given stratum (see below)
`Area`	area of the strata

Note that in the simplest case (one area surveyed only once) there is only one `Region.Label` and a single corresponding `Area` duplicated for each observation.

The example given below was provided by Eric Rexstad.

Examples

```
## Not run:
library(Distance)
# Need to have the gdata library installed from CRAN, requires a system
# with perl installed (usually fine for Linux/Mac)
library(gdata)

# Need to get the file path first
# Going to the path given in the below, one can examine the format
minke.filepath <- system.file("minke.xlsx",package="Distance")

# Load the Excel file, note that header=FALSE and we add column names after
minke <- read.xls(minke.filepath, stringsAsFactor=FALSE,header=FALSE)
```

```
names(minke) <- c("Region.Label", "Area", "Sample.Label", "Effort","distance")
# One may want to call edit(minke) or head(minke) at this point
# to examine the data format

# Due to the way the file was saved and the default behaviour in R
# for numbers stored with many decimal places (they are read as strings
# rather than numbers, see str(minke)). We must coerce the Effort column
# to numeric
minke$Effort <- as.numeric(minke$Effort)

## perform an analysis using the exact distances
pooled.exact <- ds(minke, truncation=1.5, key="hr", order=0)
summary(pooled.exact)


## Try a binned analysis
# first define the bins
dist.bins <- c(0,.214, .428,.643,.857,1.071,1.286,1.5)
pooled.binned <- ds(minke, truncation=1.5, cutpoints=dist.bins, key="hr", order=0)

# binned with stratum as a covariate
minke$stratum <- ifelse(minke$Region.Label=="North", "N", "S")
strat.covar.binned <- ds(minke, truncation=1.5, key="hr",
                         formula=~as.factor(stratum), cutpoints=dist.bins)

# Stratified by North/South
full.strat.binned.North <- ds(minke[minke$Region.Label=="North",],
                  truncation=1.5, key="hr", order=0, cutpoints=dist.bins)
full.strat.binned.South <- ds(minke[minke$Region.Label=="South",],
                     truncation=1.5, key="hr", order=0, cutpoints=dist.bins)

## model summaries
model.sel.bin <- data.frame(name=c("Pooled f(0)", "Stratum covariate",
                                   "Full stratification"),
                            aic=c(pooled.binned$ddf$criterion,
                                  strat.covar.binned$ddf$criterion,
                                  full.strat.binned.North$ddf$criterion+
                                  full.strat.binned.South$ddf$criterion))

# Note model with stratum as covariate is most parsimonious
print(model.sel.bin)

## End(Not run)
```

---

gof_ds *Goodness of fit testing and quantile-quantile plots*

Description

Formal goodness of fit testing for detection function models using Kolmogorov-Smirnov and Cramer-von Mises tests. Both tests are based on looking at the quantile-quantile plot produced by `qqplot.ddf` and deviations from the line x=y.

Usage

```
gof_ds(model, plot = TRUE, chisq = FALSE, nboot = 100, ks = FALSE,
  ...)
```

Arguments

`model`	a fitted detection function.
`plot`	if `TRUE` the Q-Q plot is plotted
`chisq`	if `TRUE` then chi-squared statistic is calculated even for models that use exact distances. Ignored for models that use binned distances
`nboot`	number of replicates to use to calculate p-values for the Kolmogorov-Smirnov goodness of fit test statistics
`ks`	perform the Kolmogorov-Smirnov test (this involves many bootstraps so can take a while)
`...`	other arguments to be passed to `ddf.gof`

Details

The Kolmogorov-Smirnov test asks the question "what's the largest vertical distance between a point and the y=x line?" It uses this distance as a statistic to test the null hypothesis that the samples (EDF and CDF in our case) are from the same distribution (and hence our model fits well). If the deviation between the y=x line and the points is too large we reject the null hypothesis and say the model doesn't have a good fit.

Rather than looking at the single biggest difference between the y=x line and the points in the Q-Q plot, we might prefer to think about all the differences between line and points, since there may be many smaller differences that we want to take into account rather than looking for one large deviation. Its null hypothesis is the same, but the statistic it uses is the sum of the deviations from each of the point to the line.

Details

Note that a bootstrap procedure is required for the Kolmogorov-Smirnov test to ensure that the p-values from the procedure are correct as the we are comparing the cumulative distribution function (CDF) and empirical distribution function (EDF) and we have estimated the parameters of the detection function. The `nboot` parameter controls the number of bootstraps to use. Set to `0` to avoid computing bootstraps (much faster but with no Kolmogorov-Smirnov results, of course).

Examples

```
## Not run:
# fit and test a simple model for the golf tee data
library(Distance)
data(book.tee.data)
tee.data<-book.tee.data$book.tee.dataframe[book.tee.data$book.tee.dataframe$observer==1,]
ds.model <- ds(tee.data,4)
```

```
# don't make plot
gof_ds(ds.model, plot=FALSE)

## End(Not run)
```

`logLik.dsmodel` *log-likelihood value for a fitted detection function*

Description

Extract the log-likelihood from a fitted detection function.

Usage

```
## S3 method for class 'dsmodel'
logLik(object, ...)
```

Arguments

`object`	a fitted detection function model object
`...`	included for S3 completeness, but ignored

Value

a numeric value giving the log-likelihood with two attributes: `"df"` the "degrees of freedom" for the model (number of parameters) and `"nobs"` the number of observations used to fit the model

Author(s)

David L Miller

Examples

```
## Not run:
library(Distance)
data(minke)
model <- ds(minke, truncation=4)
# extract the log likelihood
logLik(model)

## End(Not run)
```

minke *Simulated minke whale data*

Description

Data simulated from models fitted to 1992/1993 Southern Hemisphere minke whale data collected by the International Whaling Commission. See Branch and Butterworth (2001) for survey details (survey design is shown in figure 1(e)). Data simulated by David Borchers.

Format

`data.frame` with 99 observations of 5 variables:

`Region.Label`	stratum label (`"North"` or `"South"`)
`Area`	stratum area
`Sample.Label`	transect identifier
`Effort`	transect length
`distance`	observed distance

Details

Data are included here as both R data and as an Excel spreadsheet to illustrate the "flat file" input method. See `flatfile` for how to load this data and an example analysis.

Source

Shipped with the Distance for Windows.

References

Branch, T.A. and D.S. Butterworth (2001) Southern Hemisphere minke whales: standardised abundance estimates from the 1978/79 to 1997/98 IDCR-SOWER surveys. Journal of Cetacean Research and Management 3(2): 143-174

Hedley, S.L., and S.T. Buckland. Spatial Models for Line Transect Sampling. Journal of Agricultural, Biological, and Environmental Statistics 9, no. 2 (2004): 181-199. doi:10.1198/1085711043578.

Examples

```
data(minke)
head(minke)
```

`plot.dsmodel` *Plot a fitted detection function*

Description

This is just a simple wrapper around `plot.ds`. See the manual page for that function for more information.

Usage

```
## S3 method for class 'dsmodel'
plot(x, pl.den = 0, ...)
```

Arguments

`x`	an object of class `dsmodel`.
`pl.den`	shading density for histogram (default `0`, no shading)
`...`	extra arguments to be passed to `plot.ds`.

Value

`NULL`, just produces a plot.

Author(s)

David L. Miller

`print.dsmodel` *Simple pretty printer for distance sampling analyses*

Description

Simply prints out a brief description of the model which was fitted. For more detailed information use `summary`.

Usage

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

Arguments

`x`	a distance sampling analysis (result from calling `ds`).
`...`	not passed through, just for S3 compatibility.

Author(s)

David L. Miller

print.summary.dsmodel *Print summary of distance detection function model object*

Description

Provides a brief summary of a distance sampling analysis. Including: detection function parameters, model selection criterion, and optionally abundance in the covered (sampled) region and its standard error.

Usage

```
## S3 method for class 'summary.dsmodel'
print(x, ...)
```

Arguments

x	a summary of distance sampling analysis
...	unspecified and unused arguments for S3 consistency

Value

Nothing, just prints the summary.

Author(s)

David L. Miller and Jeff Laake

See Also

summary.ds

summarize_ds_models *Make a table of summary statistics for detection function models*

Description

Provide a summary table of useful information about fitted detection functions. This can be useful when paired with `knitrs` `kable` function. By default models are sorted by AIC and will therefore not allow models with different truncations and distance binning.

Usage

```
summarize_ds_models(..., sort = "AIC", output = "latex",
  delta_only = TRUE)
```

Arguments

`...`	models to be summarised
`sort`	column to sort by (default `"AIC"`)
`output`	should the output be given in `"latex"` compatible format or as `"plain"` text?
`delta_only`	only output AIC differences (default `TRUE`)

Details

Note that the column names are in LaTeX format, so if you plan to manipulate the resulting `data.frame` in R, you may wish to rename the columns for ease of access.

Author(s)

David L Miller

Examples

```
## Not run:
# fit some models to the golf tee data
library(Distance)
data(book.tee.data)
tee.data<-book.tee.data$book.tee.dataframe[book.tee.data$book.tee.dataframe$observer==1,]
model_hn <- ds(tee.data,4)
model_hr <- ds(tee.data,4, key="hr")
summarize_ds_models(model_hr, model_hn, output="plain")

## End(Not run)
```

`summary.dsmodel` *Summary of distance sampling analysis*

Description

Provides a brief summary of a distance sampling analysis. This includes

Usage

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

Arguments

`object`	a distance analysis
`...`	unspecified and unused arguments for S3 consistency

Value

list of extracted and summarized objects

Note

This function just calls `summary.ds` and `dht`, collates and prints the results in a nice way.

Author(s)

David L. Miller

Index