Package: Distance (via r-universe)

October 21, 2026

License GPL (>= 2)

Title Distance Sampling Detection Function and Abundance Estimation

LazyLoad yes

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. See Miller et al. (2019) <doi:10.18637/jss.v089.i01> for more information on methods and <https://distancesampling.org/resources/vignettes.html> for example analyses.

Version 2.0.1.9000

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

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

Language en-GB

Depends R (>= 4.1.0), mrds (>= 3.0.1)

Imports dplyr, methods, rlang, Rdpack

Suggests rmarkdown, kableExtra, bookdown, knitr, covr, progress, parallel, doParallel, doRNG, foreach, activity, testthat, optimx, readxl

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

RdMacros Rdpack

Config/pak/sysreqs cmake

Repository https://distancedevelopment.r-universe.dev

Date/Publication 2025-10-17 11:36:55 UTC

RemoteUrl https://github.com/distanceDevelopment/Distance

RemoteRef HEAD

RemoteSha f161b2346995481e4386d476387ff8c8e1367fbe

Contents

Distance-package . . . 3
add_df_covar_line . . . 4
AIC.dsmodel . . . 5
amakihi . . . 6
bootdht . . . 7
bootdht_Dhat_summarize . . . 10
bootdht_Nhat_summarize . . . 11
capercaillie . . . 11
checkdata . . . 12
ClusterExercise . . . 13
convert_units . . . 13
create.bins . . . 14
create_bins . . . 15
CueCountingExample . . . 16
dht2 . . . 17
ds . . . 22
ds.gof . . . 29
ducknest . . . 30
DuikerCameraTraps . . . 31
dummy_ddf . . . 31
ETP_Dolphin . . . 32
flatfile . . . 33
gof_ds . . . 35
golftees . . . 36
logLik.dsmodel . . . 37
LTExercise . . . 38
make_activity_fn . . . 39
minke . . . 40
p_dist_table . . . 41
plot.dsmodel . . . 42
predict.dsmodel . . . 43
predict.fake_ddf . . . 44
print.dht_result . . . 45
print.dsmodel . . . 45
print.summary.dsmodel . . . 46
PTExercise . . . 46
QAIC . . . 47
Savannah_sparrow_1980 . . . 49
sikadeer . . . 50
Stratify_example . . . 51
summarize_ds_models . . . 51
summary.dht_bootstrap . . . 52

summary.dsmodel 53
Systematic_variance_1 54
unflatten 54
unimak 55
units_table 56
wren 56

Index **58**

`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`.

Examples of distance sampling analyses are available at `https://distancesampling.org/resources/vignettes.html`.

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

Bugs can be reported at `https://github.com/DistanceDevelopment/Distance/issues`.

Author(s)

David L. Miller dave@ninepointeightone.net

References

"_PACKAGE"

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. doi: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.

`add_df_covar_line` *Add covariate levels detection function plots*

Description

Add a line or lines to a plot of the detection function which correspond to a a given covariate combination. These can be particularly useful when there is a small number of factor levels or if quantiles of a continuous covariate are specified.

Arguments

`ddf`	a fitted detection function object.
`data`	a `data.frame` with the covariate combination you want to plot.
`...`	extra arguments to give to `lines` (e.g., `lty`, `lwd`, `col`).
`ndist`	number of distances at which to evaluate the detection function.
`pdf`	should the line be drawn on the probability density scale; ignored for line transects
`breaks`	required to ensure that PDF lines are the right size, should match what is supplied to original `plot` command. Defaults to "Sturges" breaks, as in `hist`. Only used if `pdf=TRUE`

Details

All covariates must be specified in `data`. Plots can become quite busy when this approach is used. It may be useful to fix some covariates at their median level and plot set values of a covariate of interest. For example setting weather (e.g., Beaufort) to its median and plotting levels of observer, then creating a second plot for a fixed observer with levels of weather.

Arguments to `lines` are supplied in `...` and aesthetics like line type (`lty`), line width (`lwd`) and colour (`col`) are recycled. By default `lty` is used to distinguish between the lines. It may be useful to add a `legend` to the plot (lines are plotted in the order of `data`).

Value

invisibly, the values of detectability over the truncation range.

Note

This function is located in the `mrds` package but the documentation is provided here for easy access.

Author(s)

David L Miller

Examples

```
## Not run:
# example using a model for the minke data
data(minke)
# fit a model
result <- ds(minke, formula=~Region.Label)

# make a base plot, showpoints=FALSE makes the plot less busy
plot(result, showpoints=FALSE)

# add lines for sex one at a time
add_df_covar_line(result, data.frame(Region.Label="South"), lty=2)
add_df_covar_line(result, data.frame(Region.Label="North"), lty=3)

# add a legend
legend(1.5, 1, c("Average", "South", "North"), lty=1:3)

# point transect example
data(amakihi)
result <- ds(amakihi, truncation=150, transect="point", formula=~OBs)
plot(result, showpoints=FALSE, pdf=TRUE)
add_df_covar_line(result,
                  data.frame(OBs=na.omit(unique(amakihi$OBs))), pdf=TRUE)

## End(Not run)
```

`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` *Hawaiian amakihi point transect data*

Description

Also known as the Common 'Amakihi, a type of Hawaiian honeycreeper

Format

A `data.frame` with 1487 rows and 12 variables

- `Region.Label` strata names (seven strata)
- `Area` size of study area (set to 0)
- `Sample.Label` transect ID
- `Effort` number of visits to point
- `object` object ID
- `distance` radial distance (m)
- `Month` month survey conducted (not used)
- `OBs` observer ID (note capitalisation of variable name)
- `Sp` species code (COAM) for all detections
- `MAS` Time after sunrise (min)
- `HAS` Time after sunrise (hours)
- `Study.Area` name of study area

Note

Example for investigating covariates in the detection function. Note high colinearity between two measures of time since sunrise. Convergence problems can result from models with several factor covariates.

References

Marques, T.A., L. Thomas, S.G. Fancy and S.T. Buckland. (2007) Improving estimates of bird density using multiple-covariate distance sampling. The Auk 124 (4): 1229–1243. doi:10.1642/00048038(2007)124[1229:IEOBDU]2.0.CO;2

bootdht *Bootstrap uncertainty estimation for distance sampling models*

Description

Performs a bootstrap for simple distance sampling models using the same data structures as dht. Note that only geographical stratification as supported in dht is allowed.

Usage

```
bootdht(
  model,
  flatfile,
  resample_strata = FALSE,
  resample_obs = FALSE,
  resample_transects = TRUE,
  nboot = 100,
  summary_fun = bootdht_Nhat_summarize,
  convert_units = 1,
  select_adjustments = FALSE,
  sample_fraction = 1,
  multipliers = NULL,
  progress_bar = "base",
  cores = 1,
  convert.units = NULL
)
```

Arguments

model	a model fitted by ds or a list of models
flatfile	Data provided in the flatfile format. See flatfile for details. Please note, it is a current limitation of bootdht that all Sample.Label identifiers must be unique across all strata, i.e.transect ids must not be re-used from one strata to another. An easy way to achieve this is to paste together the stratum names and transect ids.
resample_strata	should resampling happen at the stratum (Region.Label) level? (Default FALSE)
resample_obs	should resampling happen at the observation (object) level? (Default FALSE)
resample_transects	should resampling happen at the transect (Sample.Label) level? (Default TRUE)
nboot	number of bootstrap replicates
summary_fun	function that is used to obtain summary statistics from the bootstrap, see Summary Functions below. By default bootdht_Nhat_summarize is used, which just extracts abundance estimates.

convert_units — conversion between units for abundance estimation, see "Units", below. (Defaults to 1, implying all of the units are "correct" already.) This takes precedence over any unit conversion stored in model.

select_adjustments — select the number of adjustments in each bootstrap, when FALSE the exact detection function specified in model is fitted to each replicate. Setting this option to TRUE can significantly increase the runtime for the bootstrap. Note that for this to work model must have been fitted with adjustment!=NULL.

sample_fraction — what proportion of the transects was covered (e.g., 0.5 for one-sided line transects).

multipliers — list of multipliers. See "Multipliers" below.

progress_bar — which progress bar should be used? Default "base" uses txtProgressBar, "none" suppresses output, "progress" uses the progress package, if installed.

cores — number of CPU cores to use to compute the estimates. See "Parallelization" below.

convert.units — deprecated, see same argument with underscore, above.

Summary Functions

The function summary_fun allows the user to specify what summary statistics should be recorded from each bootstrap. The function should take two arguments, ests and fit. The former is the output from dht2, giving tables of estimates. The latter is the fitted detection function object. The function is called once fitting and estimation has been performed and should return a data.frame. Those data.frames are then concatenated using rbind. One can make these functions return any information within those objects, for example abundance or density estimates or the AIC for each model. See Examples below.

Multipliers

It is often the case that we cannot measure distances to individuals or groups directly, but instead need to estimate distances to something they produce (e.g., for whales, their blows; for elephants their dung) – this is referred to as indirect sampling. We may need to use estimates of production rate and decay rate for these estimates (in the case of dung or nests) or just production rates (in the case of songbird calls or whale blows). We refer to these conversions between "number of cues" and "number of animals" as "multipliers".

The multipliers argument is a list, with 3 possible elements (creation and decay). Each element of which is either:

- data.frame and must have at least a column named rate, which abundance estimates will be divided by (the term "multiplier" is a misnomer, but kept for compatibility with Distance for Windows). Additional columns can be added to give the standard error and degrees of freedom for the rate if known as SE and df, respectively. You can use a multirow data.frame to have different rates for different geographical areas (for example). In this case the rows need to have a column (or columns) to merge with the data (for example Region.Label).
- a function which will return a single estimate of the relevant multiplier. See make_activity_fn for a helper function for use with the activity package.

Model selection

Model selection can be performed on a per-replicate basis within the bootstrap. This has three variations:

1. when `select_adjustments` is `TRUE` then adjustment terms are selected by AIC within each bootstrap replicate (provided that `model` had the `order` and `adjustment` options set to non-`NULL`.
2. if `model` is a list of fitted detection functions, each of these is fitted to each replicate and results generated from the one with the lowest AIC.
3. when `select_adjustments` is `TRUE` and `model` is a list of fitted detection functions, each model fitted to each replicate and number of adjustments is selected via AIC. This last option can be extremely time consuming.

Parallelization

If `cores`>1 then the `parallel`/`doParallel`/`foreach`/`doRNG` packages will be used to run the computation over multiple cores of the computer. To use this component you need to install those packages using: `install.packages(c("foreach", "doParallel", "doRNG"))` It is advised that you do not set `cores` to be greater than one less than the number of cores on your machine. The `doRNG` package is required to make analyses reproducible (`set.seed` can be used to ensure the same answers).

It is also hard to debug any issues in `summary_fun` so it is best to run a small number of bootstraps first in parallel to check that things work. On Windows systems `summary_fun` does not have access to the global environment when running in parallel, so all computations must be made using only its `ests` and `fit` arguments (i.e., you can not use R objects from elsewhere in that function, even if they are available to you from the console).

Another consequence of the global environment being unavailable inside parallel bootstraps is that any starting values in the model object passed in to `bootdht` must be hard coded (otherwise you get back 0 successful bootstraps). For a worked example showing this, see the camera trap distance sampling online example at https://distancesampling.org/Distance/articles/web-only/CTDS/camera-distill.html.

See Also

`summary.dht_bootstrap` for how to summarize the results, `bootdht_Nhat_summarize` and `bootdht_Dhat_summarize` for an examples of summary functions.

Examples

```
## Not run:
# fit a model to the minke data
data(minke)
mod1 <- ds(minke)

# summary function to save the abundance estimate
Nhat_summarize <- function(ests, fit) {
  return(data.frame(Nhat=ests$individuals$N$Estimate))
}
```

```
# perform 5 bootstraps
bootout <- bootdht(mod1, flatfile=minke, summary_fun=Nhat_summarize, nboot=5)

# obtain basic summary information
summary(bootout)

## End(Not run)
```

`bootdht_Dhat_summarize` *Simple summary of density results for bootstrap model*

Description

When using `bootdht` one needs to use a summary function to extract results from the resulting models per replicate. This function is the simplest possible example of such a function, that just extracts the estimated density (with stratum labels).

Usage

```
bootdht_Dhat_summarize(ests, fit)
```

Arguments

`ests`	output from `dht2`.
`fit`	fitted detection function object (unused).

Details

Further examples of such functions can be found at `https://distancesampling.org/resources/vignettes.html`.

Value

`data.frame` with two columns ("`Dhat`" and "`Label`"), giving the estimate(s) of density of individuals per stratum from each bootstrap replicate. This `data.frame` can be examined for example, with `quantile` to compute confidence intervals.

See Also

`bootdht` which this function is to be used with and `bootdht_Nhat_summarize` which does the same job but returns abundance results.

bootdht_Nhat_summarize *Simple summary of abundance results for bootstrap model*

Description

When using bootdht one needs to use a summary function to extract results from the resulting models per replicate. This function is the simplest possible example of such a function, that just extracts the estimated abundance (with stratum labels).

Usage

```
bootdht_Nhat_summarize(ests, fit)
```

Arguments

ests	output from dht2.
fit	fitted detection function object (unused).

Details

Further examples of such functions can be found at https://distancesampling.org/resources/vignettes.html.

Value

data.frame with two columns ("Nhat" and "Label"), giving the estimate(s) of abundance of individuals per stratum from each bootstrap replicate. This data.frame can be examined for example, with quantile to compute confidence intervals.

See Also

bootdht which this function is to be used with and bootdht_Dhat_summarize which does the same job but for abundance results.

capercaillie *Capercaillie in Monaughty Forest*

Description

Data from a line transect survey of capercaillie in Monaughty Forest, Moray, Scotland.

Format

A data.frame with 112 observations on the following 9 variables.

- Sample.Label name of single transect
- Effort transect length (km)
- distance perpendicular distance (m)
- object object ID
- size only individual birds detected
- detected whether detected
- observer single observer data
- Region.Label stratum name
- Area size of Monaughty Forest (ha)

---

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

ClusterExercise *Simulated minke whale data with cluster size*

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 9 variables:

- `Region.Label` stratum label (`"North"` or `"South"`)
- `Area` stratum area (square nautical mile)
- `Sample.Label` transect identifier
- `Effort` transect length (nautical mile)
- `object` unique object ID
- `distance` observed distance (nautical mile)
- `Cluster.strat` strata based on cluster size: 1, 2 and 3+
- `size` cluster size
- `Study.Area` name of study area

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. (2004) Spatial models for line transect sampling. Journal of Agricultural, Biological, and Environmental Statistics 9: 181-199. doi:10.1198/1085711043578.

convert_units *Convert units for abundance estimation*

Description

It is often the case that effort, distances and prediction area are collected in different units in the field. Functions in `Distance` allow for an argument to convert between these and provide an answer that makes sense. This function calculates that conversion factor, given knowledge of the units of the quantities used.

Usage

```
convert_units(distance_units, effort_units, area_units)
```

Arguments

distance_units	units distances were measured in.
effort_units	units that effort were measured in. Set as NULL for point transects.
area_units	units for the prediction area.

Details

convert_units expects particular names for its inputs – these should be singular names of the unit (e.g., "metre" rather than "metres"). You can view possible options with units_table. Both UK and US spellings are acceptable, case does not matter. For density estimation, area must still be provided ("objects per square ???"). Note that for cue counts (or other multiplier-based methods) one will still have to ensure that the rates are in the correct units for the survey.

Author(s)

David L Miller

Examples

```
# distances measured in metres, effort in kilometres and
# abundance over an area measured in hectares:
convert_units("Metre", "Kilometre", "Hectare")

# all SI units, so the result is 1
convert_units("Metre", "metre", "square metre")

# for points ignore effort
convert_units("Metre", NULL, "Hectare")
```

---

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

Description

create.bins is now deprecated, please use create_bins

Usage

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

Arguments

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

Value

argument data with two extra columns distbegin and distend.

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

argument `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)
```

CueCountingExample *Cue counts of whale blows*

Description

Cues are treated as an indirect count, requiring the use of multipliers.

Format

A `data.frame` with 109 rows and 15 variables.

- 'Region.Label stratum labels
- `Area` size (km^2) of each stratum
- `Sample.Label` transect labels
- `Cue.rate` rate of blows per animal per hour
- `Cue.rate.SE` variability in cue rate
- `Cue.rate.df` degrees of freedom (number of animals sampled for cues)
- `object` object ID
- `distance` perpendicular distance (km)
- `Sample.Fraction` proportion of full circle scanned (radians)
- `Sample.Fraction.SE` variability in sampling fraction (0)
- `Search.time` Duration of scanning effort (hr)
- `bss` Beaufort sea state
- `sp` Species detected (all observations W in these data)
- `size` Number of animals in group (all 1 in these data)
- `Study.Area` study area name

Details

Because whale blows disappear instantaneously, there is no need to measure a decay rate. However a cue production rate (blows per individual per unit time) is required, as is a measure of variability of that rate.

Note

There are two other nuances in this survey. Even though the survey is taking place on a moving ship, effort is measured as amount of time scanning for blows. In some instances, it is not possible for the observer to scan the sea all around them as view may be restricted by the ship's superstructure. Here a `sampling fraction` multiplier is employed to deal with restricted vision. Units of measure of `cue.rate` and `Search.time` must be equal.

dht2 *Abundance estimation for distance sampling models*

Description

Once a detection function is fitted to data, this function can be used to compute abundance estimates over required areas. The function also allows for stratification and variance estimation via various schemes (see below).

Usage

```
dht2(
  ddf,
  observations = NULL,
  transects = NULL,
  geo_strat = NULL,
  flatfile = NULL,
  strat_formula,
  convert_units = 1,
  er_est = c("R2", "P2"),
  multipliers = NULL,
  sample_fraction = 1,
  ci_width = 0.95,
  innes = FALSE,
  stratification = "geographical",
  total_area = NULL,
  binomial_var = FALSE
)
```

Arguments

ddf	model fitted by ds or ddf. Multiple detection functions can be supplied as a list.
observations	data.frame to link detection function data (indexed by object column IDs) to the transects (indexed by Sample.Label column IDs). See "Data" below.
transects	data.frame with information about samples (points or line transects). See "Data" below.
geo_strat	data.frame with information about any geographical stratification. See "Data" below.
flatfile	data in the flatfile format, see flatfile. Note that the object column (uniquely identifying the observations) is required.
strat_formula	a formula giving the stratification structure (see "Stratification" below). Currently only one level of stratification is supported.
convert_units	conversion factor between units for the distances, effort and area. See "Units" below. Can supply one per detection function in ddf.

er_est	encounter rate variance estimator to be used. See "Variance" below and varn. Can supply one per detection function in ddf.
multipliers	list of data.frames. See "Multipliers" below.
sample_fraction	proportion of the transect covered (e.g., 0.5 for one-sided line transects). May be specified as either a single number or a data.frame with 2 columns Sample.Label and fraction (if fractions are different for each transect).
ci_width	for use with confidence interval calculation (defined as 1-alpha, so the default 95 will give a 95% confidence interval).
innes	logical flag for computing encounter rate variance using either the method of Innes et al. (2002) where estimated abundance per transect divided by effort is used as the encounter rate, vs. (when innes=FALSE) using the number of observations divided by the effort (as in Buckland et al. 2001)
stratification	what do strata represent, see "Stratification" below.
total_area	for options stratification="effort_sum" and stratification="replicate" the area to use as the total for combined, weighted final estimates.
binomial_var	if we wish to estimate abundance for the covered area only (i.e., study area = surveyed area) then this must be set to be TRUE and use the negative binomial variance estimator of Borchers et al. (1998). This is only valid when objects are not clustered. (This situation is rare.)

Value

a data.frame (of class dht_result for pretty printing) with estimates and attributes containing additional information, see "Outputs" for information on column names.

Data

The data format allows for complex stratification schemes to be set-up. Three objects are always required:

- ddf the detection function (see ds or ddf for information on the format of their inputs).
- observations has one row per observation and links the observations to the transects. Required columns:
  - object (unique ID for the observation, which must match with the data in the detection function)
  - Sample.Label (unique ID for the transect).
  - Additional columns for strata which are not included in the detection function are required (stratification covariates that are included in the detection function do not need to be included here). The important case here is group size, which must have column name size (but does not need to be in the detection function).
- transects has one row per sample (point or line transect). At least one row is required. Required columns: Sample.Label (unique ID for the transect), Effort (line length for line transects, number of visits for point transects), if there is more than one geographical stratum.

With only these three arguments, abundance can only be calculated for the covered area. Including additional information on the area we wish to extrapolate to (i.e., the study area), we can obtain abundance estimates:

- `geo_strat` has one row for each stratum that we wish to estimate abundance for. For abundance in the study area, at least one row is required. Required columns: `Area` (the area of that stratum). If there is >1 row, then additional columns, named in `strat_formula`.‘

Note that if the `Area` column is set to all 0, then only density estimates will be returned.

Multipliers

It is often the case that we cannot measure distances to individuals or groups directly, but instead need to estimate distances to something they produce (e.g., for whales, their blows; for elephants their dung) – this is referred to as indirect sampling. We may need to use estimates of production rate and decay rate for these estimates (in the case of dung or nests) or just production rates (in the case of songbird calls or whale blows). We refer to these conversions between "number of cues" and "number of animals" as "multipliers".

The `multipliers` argument is a `list`, with 2 possible elements (`creation` and `decay`). Each element of which is a `data.frame` and must have at least a column named `rate`, which abundance estimates will be divided by (the term "multiplier" is a misnomer, but kept for compatibility with Distance for Windows). Additional columns can be added to give the standard error and degrees of freedom for the rate if known as `SE` and `df`, respectively. You can use a multirow `data.frame` to have different rates for different geographical areas (for example). In this case the rows need to have a column (or columns) to `merge` with the data (for example `Region.Label`).

Stratification

The `strat_formula` argument is used to specify a column to use to stratify the results, using the form `~column.name` where `column.name` is the column name you wish to use.

The `stratification` argument is used to specify which of four types of stratification are intended:

- `"geographical"` if each stratum represents a different geographical areas and you want the total over all the areas
- `"effort_sum"` if your strata are in fact from replicate surveys (perhaps using different designs) but you don’t have many replicates and/or want an estimate of "average variance"
- `"replicate"` if you have replicate surveys but have many of them, this calculates the average abundance and the variance between those many surveys (think of a population of surveys)
- `"object"` if the stratification is really about the type of object observed, for example sex, species or life stage and what you want is the total number of individuals across all the classes of objects. For example, if you have stratified by sex and have males and females, but also want a total number of animals, you should use this option.

A simple example of using `stratification="geographical"` is given below. Further examples can be found at https://distancesampling.org/resources/vignettes.html (see, e.g., the deer pellet survey).

Variance

Variance in the estimated abundance comes from multiple sources. Depending on the data used to fit the model and estimate abundance, different components will be included in the estimated variances. In the simplest case, the detection function and encounter rate variance need to be combined. If group size varies, then this too must be included. Finally, if multipliers are used and have corresponding standard errors given, this are also included. Variances are combined by assuming independence between the measures and adding variances. A brief summary of how each component is calculated is given here, though see references for more details.

- *detection function*: variance from the detection function parameters is transformed to variance about the abundance via a sandwich estimator (see e.g., Appendix C of Borchers et al. (2002)).
- *encounter rate*: for strata with >1 transect in them, the encounter rate estimators given in Fewster et al. (2009) can be specified via the `er_est` argument. If the argument `innes=TRUE` then calculations use the estimated number of individuals in the transect (rather than the observed), which was given by Innes et al. (2002) as a superior estimator. When there is only one transect in a stratum, Poisson variance is assumed. Information on the Fewster encounter rate variance estimators are given in `varn`
- *group size*: if objects occur in groups (sometimes "clusters"), then the empirical variance of the group sizes is added to the total variance.
- *multipliers*: if multipliers with standard errors are given, their corresponding variances are added. If no standard errors are supplied, then their contribution to variance is assumed to be 0.

Units

It is often the case that distances are recorded in one convenient set of units, whereas the study area and effort are recorded in some other units. To ensure that the results from this function are in the expected units, we use the `convert_units` argument to supply a single number to convert the units of the covered area to those of the study/stratification area (results are always returned in the units of the study area). For line transects, the covered area is calculated as `2 * width * length` where `width` is the effective (half)width of the transect (often referred to as w in the literature) and `length` is the line length (referred to as L). If `width` and `length` are measured in kilometres and the study area in square kilometres, then all is fine and `convert_units` is 1 (and can be ignored). If, for example, line length and distances were measured in metres, we instead need to convert this to be kilometres, by dividing by 1000 for each of distance and length, hence `convert_units=1e-6`. For point transects, this is slightly easier as we only have the radius and study area to consider, so the conversion is just such that the units of the truncation radius are the square root of the study area units.

Output

On printing the output from call to `dht2`, three tables are produced. Below is a guide to the output columns names, per table.

- Summary statistics table
  - `Region.Label` Stratum name (this first column name depends on the `formula` supplied)
  - `Area` Size of stratum
  - `CoveredArea` Surveyed area in stratum (2 x w x L)

  - `Effort` Transect length or number of point visits per stratum
  - `n` Number of detections
  - `k` Number of replicate transects
  - `ER` Encounter rate
  - `se.ER` Standard error of encounter rate
  - `cv.ER` Coefficient of variation of encounter rate
- Abundance or density estimates table:
  - `Region.Label` As above
  - `Estimate` Point estimate of abundance or density
  - `se` Standard error
  - `cv` Coefficient of variation
  - `LCI` Lower confidence bound
  - `UCI` Upper confidence bound
  - `df` Degrees of freedom used for confidence interval computation
- Components percentage of variance:
  - `Region.Label` As above
  - `Detection` Percent of variance in abundance/density associated with detection function uncertainty
  - `ER` Percent of variance in abundance/density associated with variability in encounter rate
  - `Multipliers` Percent of variance in abundance/density associated with uncertainty in multipliers

References

Borchers DL, Buckland ST, Goedhart PW, Clarke ED, Hedley SL (1998). "Horvitz-Thompson Estimators for Double-Platform Line Transect Surveys." *Biometrics*, **54**(4), 1221-1237. doi:10.2307/253365.

Borchers DL, Buckland ST, Zucchini W (2002). *Estimating animal abundance: closed populations*. Springer.

Buckland ST, Anderson DR, Burnham KP, Laake JL, Borchers DL, Thomas L (2001). *Introduction to distance sampling: estimating abundance of biological populations*. Oxford university press.

Fewster RM, Buckland ST, Burnham KP, Borchers DL, Jupp PE, Laake JL, Thomas L (2009). "Estimating the encounter rate variance in distance sampling." *Biometrics*, **65**(1), 225-236.

Innes S, Heide-Jørgensen MP, Laake JL, Laidre KL, Cleator HJ, Richard P, Stewart RE (2002). "Surveys of belugas and narwhals in the Canadian High Arctic in 1996." *NAMMCO Scientific Publications*, **4**, 169-190.

Examples

```
## Not run:
# example of simple geographical stratification
# minke whale data, with 2 strata: North and South
```

```
data(minke)
# first fitting the detection function
minke_df <- ds(minke, truncation=1.5, adjustment=NULL)
# now estimate abundance using dht2
# stratum labels are in the Region.Label column
minke_dht2 <- dht2(minke_df, flatfile=minke, stratification="geographical",
                   strat_formula=~Region.Label)
# could compare this to minke_df$dht and see the same results
minke_dht2
# can alternatively report density
print(minke_dht2, report="density")

## 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(data$distend), ifelse(is.null(cutpoints),
    max(data$distance), max(cutpoints)), max(data$distend)),
  transect = "line",
  formula = ~1,
  key = c("hn", "hr", "unif"),
  adjustment = c("cos", "herm", "poly"),
  nadj = NULL,
  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,
  er_var = ifelse(transect == "line", "R2", "P2"),
  method = "nlminb",
  mono_method = "slsqp",
  quiet = FALSE,
```

```
  debug_level = 0,
  initial_values = NULL,
  max_adjustments = 5,
  er_method = 2,
  dht_se = TRUE,
  optimizer = "both",
  winebin = NULL,
  conf_level = 0.95,
  alpha,
  dht.group,
  region.table,
  sample.table,
  obs.table,
  convert.units,
  er.var,
  debug.level,
  initial.values,
  max.adjustments
)
```

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%","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. A value of NULL indicates that no adjustments are to be fitted.
nadj	the number of adjustment terms to fit. In the absence of covariates in the formula, the default value (NULL) will select via AIC (using a sequential forward selection algorithm) up to max.adjustment adjustments (unless order is specified). When covariates are present in the model formula, the default value of NULL results in no adjustment terms being fitted in the model. A non-negative

integer value will cause the specified number of adjustments to be fitted. Supplying an integer value will allow the use of adjustment terms in addition to specifying covariates in the model. The order of adjustment terms used will depend on the keyand adjustment. For key="unif", adjustments of order 1, 2, 3, ... are fitted when adjustment = "cos" and order 2, 4, 6, ... otherwise. For key="hn" or "hr" adjustments of order 2, 3, 4, ... are fitted when adjustment = "cos" and order 4, 6, 8, ... otherwise. See Buckland et al. (2001) p. 47 for details.

order — order of adjustment terms to fit. The default value (NULL) results in ds choosing the orders to use - see nadj. Otherwise a scalar positive integer value can be used to fit a single adjustment term of the specified order, and a vector of positive integers to fit multiple adjustment terms of the specified orders. For simple and Hermite polynomial adjustments, only even orders are allowed. The number of adjustment terms specified here must match nadj (or nadj can be the default NULL value).

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. Supplying a distance column in your data and specifying cutpoints is the recommended approach for all standard binned analyses. 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.) Provide distbegin and distend columns in your data only when your cutpoints are not constant across all your data, e.g. planes flying at differing altitudes then do not specify the cutpoints argument.

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. If Area is omitted density estimates only are produced.

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.)

`er_var` specifies which encounter rate estimator to use in the case that dht_se is TRUE, er_method is either 1 or 2 and there are two or more samplers. Defaults to "R2" for line transects and "P2" for point transects (>= 1.0.9, earlier versions <= 1.0.8 used the "P3" estimator by default for points), both of which assume random placement of transects. For systematic designs, alternative estimators may be more appropriate, see `dht2` for more information.

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

`mono_method` optimization method to use when monotonicity is enforced. Can be either `slsqp` or `solnp`. Defaults to `slsqp`.

`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`.

`er_method` encounter rate variance calculation: default = 2 gives the method of Innes et al. (2002), using expected counts in the encounter rate. Setting to 1 gives observed counts (which matches Distance for Windows) and 0 uses negative binomial variance (only useful in the rare situation where study area = surveyed area). See `dht.se` for more details, noting this `er_method` argument corresponds to the `varflag` element of the `options` argument in `dht.se`.

`dht_se` should uncertainty be calculated when using `dht`? Safe to leave as `TRUE`, used in `bootdht`.

`optimizer` By default this is set to 'both'. In this case the R optimizer will be used and if present the MCDS optimizer will also be used. The result with the best likelihood value will be selected. To run only a specified optimizer set this value to either 'R' or 'MCDS'. See `mcds_dot_exe` for setup instructions.

`winebin` If you are trying to use our MCDS.exe optimizer on a non-windows system then you may need to specify the winebin. Please see `mcds_dot_exe` for more details.

`conf_level` level of confidence used in computations of confidence interval width: default 0.95, must be between 0 and 1

`alpha` provides an alternative way to specify `conf_level`. If `alpha` is specified then `conf_level` is set to 1-`alpha`. (This takes precedence over whatever value is given to `conf_level`.)

`dht.group` deprecated, see same argument with underscore, above.

`region.table` deprecated, see same argument with underscore, above.

sample.table	deprecated, see same argument with underscore, above.
obs.table	deprecated, see same argument with underscore, above.
convert.units	deprecated, see same argument with underscore, above.
er.var	deprecated, see same argument with underscore, above.
debug.level	deprecated, see same argument with underscore, above.
initial.values	deprecated, see same argument with underscore, above.
max.adjustments	deprecated, see same argument with underscore, above.

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. Density and abundance estimates, and corresponding estimates of variance and confidence intervals, are calculated using the methods described in Buckland et al. (2001) sections 3.6.1 and 3.7.1 (further details can be found in the documentation for dht).

For more advanced abundance/density estimation please see the dht and dht2 functions.

Examples of distance sampling analyses are available at https://distancesampling.org/resources/vignettes.html.

Hints and tips on fitting (particularly optimisation issues) are on the mrds_opt manual page.

Clusters/groups

Note that if the data contains a column named size, 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 centre line 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`.

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` argument. 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 kilometres and `Area` in square kilometres and `distance` in metres then using `convert_units=0.001` would convert metres to kilometres, density would be expressed in square kilometres 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 kilometres, `Area` in hectares (100m x 100m) and `distance` in metres, then using `convert_units=10` will convert `a` to units of hectares (100 to convert metres to 100 metres 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.

Density estimation

If column `Area` is omitted, a density estimate is generated but note that the degrees of freedom/standard errors/confidence intervals will not match density estimates made with the `Area` column present.

Author(s)

David L. Miller

References

Buckland ST, Anderson DR, Burnham KP, Laake JL, Borchers DL, Thomas L (2001). *Introduction to distance sampling: estimating abundance of biological populations*. Oxford university press.

Innes S, Heide-Jørgensen MP, Laake JL, Laidre KL, Cleator HJ, Richard P, Stewart RE (2002). "Surveys of belugas and narwhals in the Canadian High Arctic in 1996." *NAMMCO Scientific Publications*, **4**, 169-190.

See Also

`flatfile`, `AIC.ds`, `ds.gof`, `p_dist_table`, `plot.ds`, `add_df_covar_line`

Examples

```
# An example from mrds, the golf tee data.
library(Distance)
data(book.tee.data)
tee.data <- subset(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

This function is deprecated, please see `gof_ds`.

Usage

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

Arguments

`model`	deprecated.
`breaks`	deprecated.
`nc`	deprecated.
`qq`	deprecated.
`ks`	deprecated.
`...`	deprecated.

Value

Nothing, deprecated.

Author(s)

David L Miller

See Also

qqplot.ddf ddf.gof

---

ducknest *Ducknest line transect survey data*

Description

Simulated line transect survey of duck nests, designed to reproduce the data of Figure 2 in Anderson and Pospahala (1970).

Format

A data.frame with 534 rows and 7 variables

- Region.Label strata names (single stratum in this instance)
- Area size of refuge (0 in this case, actual size 60km^2)
- Sample.Label transect ID
- Effort length of transects (km)
- object nest ID
- distance perpendicular distance (m)
- Study.Area name of wildlife refuge

Details

The Monte Vista National Wildlife Refuge is in southern Colorado in the USA at an altitude of roughly 2400m.

Source

Simulated data, from the distance sampling introductory course, Centre for Research into Ecological & Environmental Modelling, University of St Andrews.

References

Anderson, D. R., and R. S. Pospahala. 1970. Correction of bias in belt transect studies of immotile objects. The Journal of Wildlife Management 34 (1): 141–146. doi:10.2307/3799501

DuikerCameraTraps *Duiker camera trap survey*

Description

Study took place in Tai National Park Cote d'Ivoire in 2014. Filmed Maxwell's duikers (Philantomba maxwellii) were assigned to distance intervals; recorded distances are the midpoints of the intervals. This data includes only observations recorded at times of peak activity.

Format

A `data.frame` with 6277 rows and 6 variables

- `Region.Label` strata names (single stratum)
- `Area` size of study area (40.37 km^2)
- `multiplier` spatial effort, as the proportion of a circle covered by the angle of view of the camera (42 degrees for these cameras)
- `Sample.Label` camera station identifier (21 functioning cameras in this data set)
- `Effort` temporal effort, i.e. the number of 2-second time-steps over which the camera operated
- `object` unique object ID
- `distance` radial distance (m) to interval midpoint

Source

Howe, E.J., Buckland, S.T., Després-Einspenner, M.-L. and Kühl, H.S. (2017), Distance sampling with camera traps. Methods Ecol Evol, 8: 1558-1565. doi:10.1111/2041210X.12790

Howe, Eric J. et al. (2018), Data from: Distance sampling with camera traps, Dryad, Dataset, doi:10.5061/dryad.b4c70

dummy_ddf *Detection function objects when detection is certain*

Description

Create a detection function object for strip/plot surveys for use with `dht2`.

Usage

```
dummy_ddf(data, width, left = 0, transect = "line")
```

Arguments

data	as specified for ds and ddf (including a size column)
width	right truncation
left	left truncation (default 0, no left truncation)
transect	"line" or "point" transect

Author(s)

David L Miller

ETP_Dolphin *Eastern Tropical Pacific spotted dolphin survey*

Description

Observers aboard tuna vessels detecting dolphin schools along with a number of possibly useful covariates for modelling the detection function.

Format

A data.frame with 1090 rows and 13 variables:

- Region.Label stratum labels (only one)
- Area size (nmi) of each stratum
- Sample.Label transect labels
- Effort transect length (nmi)
- object object ID
- distance perpendicular distance (nmi)
- LnCluster natural log of cluster size
- Month month of detection
- Beauf.class Beaufort sea state
- Cue.type initial cue triggering detection
- Search.method observer method making the detection
- size cluster size
- Study.Area study area name

Details

Several different search methods included in these data

- 0 binoculars from crows nest
- 2 binoculars from elsewhere on ship
- 3 helicopter searching ahead of ship
- 5 radar detects of seabirds above dolphin schools

Several cue types were also recorded by observers.

- 1 seabirds above the school
- 2 water splashes
- 3 unspecified
- 4 floating objects such as logs

Source

Inter-American Tropical Tuna Commission

`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 `readxl::read_xls` in or CSV using `read.csv`.

Details

Each row of the data table corresponds to either: (1) an observation or (2) a sample (transect) without observations. In either case the following columns must be present:

- `distance` observed distance to object
- `object` a unique identifier for each observation (only required when using `dht2`)
- `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 strataWhen the row represents a transect without observations,distanceand any other obse

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. Additional examples can be found at https://distancesampling.org/resources/vignettes.html.

Examples

```
## Not run:
library(Distance)
# Need to have the readxl package installed from CRAN
require(readxl)

# Need to get the file path first
minke.filepath <- system.file("minke.xlsx", package="Distance")

# Load the Excel file, note that col_names=FALSE and we add column names after
minke <- read_xlsx(minke.filepath, col_names=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

## 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

Goodness of fit testing for detection function models. For continuous distances Kolmogorov-Smirnov and Cramer-von Mises tests can be used, when binned or continuous distances are used a χ^2 test can be used.

Usage

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

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)
nc	number of evenly-spaced distance classes for chi-squared test, if chisq=TRUE
breaks	vector of cutpoints to use for binning, if chisq=TRUE
...	other arguments to be passed to ddf.gof

Details

Kolmogorov-Smirnov and Cramer-von Mises tests are based on looking at the quantile-quantile plot produced by qqplot.ddf and deviations from the line $x = y$.

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.

A chi-squared test is also run if `chisq=TRUE`. In this case binning of distances is required if distance data are continuous. This can be specified as a number of equally-spaced bins (using the argument `nc=`) or the cutpoints of bins (using `breaks=`). The test compares the number of observations in a given bin to the number predicted under the fitted detection function.

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 <- subset(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)
```

---

golftees *Golf tee data*

Description

The data are from independent surveys by eight observers of a population of 250 groups (760 individuals) of golf tees. The tees, of two colours, were placed in groups of between 1 and 8 in a survey region of 1680 m^2, either exposed above the surrounding grass, or at least partially hidden by it. They were surveyed by the 1999 statistics honours class at the University of St Andrews.

Format

Data is a `list` with 4 elements each of which is a `data.frame`:

- `book.tee.dataframe`
  - `object` object ID
  - `observer` observer ID

  - detected detected or not detected
  - distance perpendicular distance
  - size group size
  - sex number of tees in group
  - exposure tee height above ground
- book.tee.region
  - Region.Label stratum name
  - Area stratum size
- book.tee.samples
  - Sample.Label transect label
  - Region.Label stratum name
  - Effort transect length
- book.tee.obs
  - object object ID
  - Region.Label stratum in which it was detected
  - Sample.Label transect on which it was detected

Details

We treat each group of golf tees as a single animal with size equal to the number of tees in the group; yellow tees are male, green are female; tees exposed above the surrounding grass are classified as exposed, others as unexposed. We are grateful to Miguel Bernal for making these data available; they were collected by him as part of a masters project.

References

Borchers, D. L., S.T. Buckland, and W. Zucchini. 2002. Estimating Animal Abundance: Closed Populations. Statistics for Biology and Health. London: Springer-Verlag. https://link.springer.com/book/10.1007/978-1-4471-3708-5

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

---

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)
```

LTExercise *Simulated line transect survey data*

Description

Simulated line transect survey. Twelve transects, detection function is half-normal. True object density is 79.8 animals per km^2.

Format

A data.frame with 106 rows and 7 variables

- Region.Label strata names (single stratum)
- Area size of study area (1 in this case, making abundance and density equal)
- Sample.Label transect ID
- Effort length of transects (km)
- object object ID
- distance perpendicular distance (m)
- Study.Area name of study area

Note

There is no unit object associated with this dataset

Source

Simulated data, from the distance sampling introductory course, Centre for Research into Ecological & Environmental Modelling, University of St Andrews.

`make_activity_fn` *Multiplier bootstrap helper functions*

Description

Helper to use a models specified using `activity::fitact` to fit an activity model and generate single realisations for bootstrapping with `bootdht`.

Usage

```
make_activity_fn(..., detector_daily_duration = 24)
```

Arguments

`...`	parameters specified by activity::fitact
`detector_daily_duration`	by default we assume that detectors were able to detect animals for 24 hours, if they were only able to do this for some proportion of the day (say daylight hours), then adjust this argument accordingly

Details

Uses `activity::fitact` to generate single possible availability estimates based on bootstraps. The function returns another function, which can be passed to `bootdht`. It is recommended that you try out the function before passing it to `bootdht`. See examples for a template for use.

Value

a function which generates a single bootstrap estimate of availability

Author(s)

David L Miller

`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
- `object` unique object ID

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)
```

p_dist_table *Distribution of probabilities of detection*

Description

Generate a table of frequencies of probability of detection from a detection function model. This is particularly useful when employing covariates, as it can indicate if there are detections with very small detection probabilities that can be unduly influential when calculating abundance estimates.

Arguments

object	fitted detection function
bins	how the results should be binned
proportion	should proportions be returned as well as counts?

Details

Because dht uses a Horvitz-Thompson-like estimator, abundance estimates can be sensitive to errors in the estimated probabilities. The estimator is based on $\sum 1/\hat{P}_a(z_i)$, which means that the sensitivity is greater for smaller detection probabilities. As a rough guide, we recommend that the method be not used if more than say 5% of the $\hat{P}_a(z_i)$ are less than 0.2, or if any are less than 0.1. If these conditions are violated, the truncation distance w can be reduced. This causes some loss of precision relative to standard distance sampling without covariates.

Value

a data.frame with probability bins, counts and (optionally) proportions. The object has an attribute p_range which contains the range of estimated detection probabilities

Note

This function is located in the mrds package but the documentation is provided here for easy access.

Author(s)

David L Miller

References

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.

Examples

```
# example using a model for the minke data
data(minke)
# fit a model
result <- ds(minke, formula=~Region.Label)
# print table
p_dist_table(result)
# with proportions
p_dist_table(result, proportion=TRUE)
```

`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

See Also

`add_df_covar_line`

`predict.dsmodel` *Predictions from a fitted detection function*

Description

Predict detection probabilities (or effective strip widths/effective areas of detection) from a fitted distance sampling model using either the original data (i.e., "fitted" values) or using new data.

Usage

```
## S3 method for class 'dsmodel'
predict(
  object,
  newdata = NULL,
  compute = FALSE,
  esw = FALSE,
  se.fit = FALSE,
  ...
)
```

Arguments

`object`	`ds` model object.
`newdata`	new `data.frame` for prediction, this must include a column called "`distance`".
`compute`	if `TRUE` compute values and don't use the fitted values stored in the model object.
`esw`	if `TRUE`, returns effective strip half-width (or effective area of detection for point transect models) integral from 0 to the truncation distance (`width`) of $p(y)dy$; otherwise it returns the integral from 0 to truncation width of $p(y)\pi(y)$ where $\pi(y) = 1/w$ for lines and $\pi(y) = 2r/w^2$ for points.
`se.fit`	should standard errors on the predicted probabilities of detection (or ESW if `esw=TRUE`) estimated? Stored in the `se.fit` element
`...`	for S3 consistency

Details

For line transects, the effective strip half-width (`esw=TRUE`) is the integral of the fitted detection function over either 0 to W or the specified `int.range`. The predicted detection probability is the average probability which is simply the integral divided by the distance range. For point transect models, `esw=TRUE` calculates the effective area of detection (commonly referred to as "nu", this is the integral of `2/width^2 * r * g(r)`.

Fitted detection probabilities are stored in the `model` object and these are returned unless `compute=TRUE` or `newdata` is specified. `compute=TRUE` is used to estimate numerical derivatives for use in delta method approximations to the variance.

Note that the ordering of the returned results when no new data is supplied (the "fitted" values) will not necessarily be the same as the data supplied to `ddf`, the data (and hence results from `predict`) will be sorted by object ID (`object`).

Value

a list with a single element: `fitted`, a vector of average detection probabilities or esw values for each observation in the original data or`newdata`. If `se.fit=TRUE` there is an additional element `$se.fit`, which contains the standard errors of the probabilities of detection or ESW.

Author(s)

David L Miller

`predict.fake_ddf` *Prediction for fake detection functions*

Description

Prediction function for dummy detection functions. The function returns as many 1s as there are rows in `newdata`. If `esw=TRUE` then the strip width is returned.

Usage

```
## S3 method for class 'fake_ddf'
predict(
  object,
  newdata = NULL,
  compute = FALSE,
  int.range = NULL,
  esw = FALSE,
  ...
)
```

Arguments

`object`	model object
`newdata`	how many 1s should we return?
`compute`	unused, compatibility with `mrds::predict`
`int.range`	unused, compatibility with `mrds::predict`
`esw`	should the strip width be returned?
`...`	for S3 consistency

Author(s)

David L Miller

print.dht_result *Print abundance estimates*

Description

See dht2 for information on printed column names.

Usage

```
## S3 method for class 'dht_result'
print(x, report = "abundance", groups = TRUE, ...)
```

Arguments

x	object of class dht_result
report	should "abundance", "density" or "both" be reported?
groups	should abundance/density of groups be produced? TRUE by default.
...	unused

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

PTExercise *Simulated point transect survey data*

Description

Simulated point transect survey. Thirty point transects, detection function is half-normal. True object density is 79.6 animals per hectare.

Format

A `data.frame` with 144 rows and 7 variables

- `Region.Label` strata names (single stratum)
- `Area` size of study area (0 in this case)
- `Sample.Label` transect ID
- `Effort` number of visits to point
- `object` object ID
- `distance` radial distance (m)
- `Study.Area` name of study area

Source

Simulated data, from the distance sampling introductory course, Centre for Research into Ecological & Environmental Modelling, University of St Andrews.

`QAIC` *Tools for model selection when distance sampling data are overdispersed*

Description

Overdispersion causes AIC to select overly-complex models, so analysts should specify the number/order of adjustment terms manually when fitting distance sampling models to data from camera traps, rather than allowing automated selection using AIC. Howe et al (2019) described a two-step method for selecting among models of the detection function in the face of overdispersion.

Usage

```
QAIC(object, ..., chat = NULL, k = 2)

chi2_select(object, ...)
```

Arguments

`object`	a fitted detection function object
`...`	additional fitted model objects.
`chat`	a value of $\hat{c}$ to be used in QAIC calculation
`k`	penalty per parameter to be used; default 2

Details

In step 1, and overdispersion factor ($\hat{c}$) is computed either (1) for each key function family, from the most complex model in each family, as the chi-square goodness of fit test statistic divided by its degrees of freedom ($\hat{c}_1$), or (2) for all models in the candidate set, from the raw data ($\hat{c}_1$). In camera trap surveys of solitary animals, $\hat{c}_1$ would be the mean number of distance observations recorded during a single pass by an animal in front of a trap. In surveys of social animals employing human observers, $\hat{c}_1$ would be the mean number of detected animals per detected group, and in camera trap surveys of social animals $\hat{c}_1$ the mean number of distance observations recorded during an encounter between a group of animals and a CT. In step two, the chi-square goodness of fit statistic divided by its degrees of freedom is calculated for the QAIC-minimizing model within each key function, and the model with the lowest value is selected for estimation.

The `QAIC()` function should only be used select among models with the same key function (step 1). `QAIC()` uses $\hat{c}_1$ by default, computing it from the model with the most parameters. Alternatively, $\hat{c}_1$ can be calculated from the raw data and included in the call to `QAIC()`. Users must identify the QAIC-minimizing model within key functions from the resulting `data.frame`, and provide these models as arguments in `ch2_select()`. `chi2_select()` then computes and reports the chi-square goodness of fit statistic divided by its degrees of freedom for each of those models. The model with the lowest value is recommended for estimation.

Value

a `data.frame` with one row per model supplied, in the same order as given

Author(s)

David L Miller, based on code from Eric Rexstad and explanation from Eric Howe.

References

Howe, E. J., Buckland, S. T., Després-Einspenner, M.-L., & Kühl, H. S. (2019). Model selection with overdispersed distance sampling data. Methods in Ecology and Evolution, 10(1), 38–47. doi:10.1111/2041210X.13082

Examples

```
## Not run:
library(Distance)
data("wren_cuecount")

# fit hazard-rate key models
w3.hr0 <- ds(wren_cuecount, transect="point", key="hr", adjustment=NULL,
             truncation=92.5)
w3.hr1 <- ds(wren_cuecount, transect="point", key="hr", adjustment="cos",
             order=2, truncation=92.5)
w3.hr2 <- ds(wren_cuecount, transect="point", key="hr", adjustment="cos",
             order=c(2, 4), truncation=92.5)

# fit unform key models
w3.u1 <- ds(wren_cuecount, transect="point", key="unif", adjustment="cos",
            order=1, truncation=92.5)
```

```
w3.u2 <- ds(wren_cuecount, transect="point", key="unif", adjustment="cos",
            order=c(1,2), monotonicity="none",  truncation=92.5)
w3.u3 <- ds(wren_cuecount, transect="point", key="unif", adjustment="cos",
            order=c(1,2,3), monotonicity="none", truncation=92.5)

# fit half-normal key functions
w3.hn0 <- ds(wren_cuecount, transect="point", key="hn", adjustment=NULL,
             truncation=92.5)
w3.hn1 <- ds(wren_cuecount, transect="point", key="hn", adjustment="herm",
             order=2, truncation=92.5)

# stage 1: calculate QAIC per model set
QAIC(w3.hr0, w3.hr1, w3.hr2)  # no adjustments smallest
QAIC(w3.u1, w3.u2, w3.u3)     # 2 adjustment terms (by 0.07)
QAIC(w3.hn0, w3.hn1)  # no adjustments smallest

# stage 2: select using chi^2/degrees of freedom between sets
chi2_select(w3.hr0, w3.u2, w3.hn0)

# example using a pre-calculated chat
chat <- attr(QAIC(w3.hr0, w3.hr1, w3.hr2), "chat")
QAIC(w3.hr0, chat=chat)

## End(Not run)
```

`Savannah_sparrow_1980` *Savanna sparrow point transects*

Description

Point transect data collected in Colorado 1980/81 to examine effect of agricultural practices upon avian community.

Format

`data.frame` with 468 observations (1980) and 448 observations (1981) of 7 variables:

- `Region.Label` stratum label (pasture ID)
- `Area` stratum area (set to 1 so density is reported)
- `Sample.Label` transect identifier
- `Effort` number of visits
- `object` object ID
- `distance` radial distance (m)
- `Study.Area` name of study area

Details

Design consisted of point transects placed in multiple pastures (3 in 1980 and 4 in 1981). While many species were observed, only data for Savannah sparrows (*Passerculus sandwichensis*) are included here.

Data given here are different from the Distance for Windows example project. Here each individual sighting is treated as an independent observation. This corresponds to the analysis in Buckland et al. (2001) Section 8.7. In the Distance for Windows project objects are clusters of individuals. This should not affect the results too greatly as most clusters were of size 1, and so the results obtained should not be too far out.

References

Knopf, F.L., J.A. Sedgwick, and R.W. Cannon. (1988) Guild structure of a riparian avifauna relative to seasonal cattle grazing. The Journal of Wildlife Management 52 (2): 280–290. doi:10.2307/3801235

---

sikadeer *Sika deer pellet data from southern Scotland*

Description

Because sika deer spend most of their time in woodland areas, abundance estimates are based on pellet group counts. Line transect methods were applied to estimate deer pellet group density by geographic block.

Format

A `data.frame` with 1923 rows and 11 variables.

- `Region.Label` stratum labels
- `Area` size (ha) of each stratum
- `Sample.Label` transect labels
- `Defecation.rate` rate of dung production per individual per day
- `Defecation.rate.SE` variability in defecation rate
- `Decay.rate` time (days) for dung to become undetectable
- `Decay.rate.SE` variability in decay rate
- `Effort` transect length (km)
- `object` object ID
- `distance` perpendicular distance (cm)
- `Study.Area` study area name

Details

Data presented here are from the Peebleshire portion of the study described by Marques et al. (2001).

References

Marques, F.F.C., S.T. Buckland, D. Goffin, C.E. Dixon, D.L. Borchers, B.A. Mayle, and A.J. Peace. (2001). Estimating deer abundance from line transect surveys of dung: sika deer in southern Scotland. Journal of Applied Ecology 38 (2): 349–363. doi:10.1046/j.13652664.2001.00584.x

---

`Stratify_example` *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 7 variables: `Region.Label` stratum label (`"North"` or `"South"`) `Area` stratum area (square nautical mile) `Sample.Label` transect identifier `Effort` transect length (nautical mile) `object` object ID `distance` observed distance (nautical mile) `Study.Area` name of study area

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. (2004) Spatial models for line transect sampling. Journal of Agricultural, Biological, and Environmental Statistics 9: 181-199. doi:10.1198/1085711043578.

`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 `knitr`'s `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(
  ...,
  models = list(),
  sort = "AIC",
  output = "latex",
  delta_only = TRUE
)
```

Arguments

...	models to be summarised (to be deprecated)
models	a named list of models to be summarised. If the list is not named then default names of 'model 1', 'model 2' etc. are used.
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 <- subset(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.dht_bootstrap *Summarize bootstrap abundance uncertainty estimate output*

Description

A simple function to calculate summaries of bootstrap output generated by bootdht.

Usage

```
## S3 method for class 'dht_bootstrap'
summary(object, alpha = 0.05, ...)
```

Arguments

object	output from bootdht
alpha	value to use in confidence interval calculation (to obtain alpha/2 and 1-alpha/2 intervals
...	for S3 compatibility, unused.

Details

Summaries are only made for numeric outputs. Both median and mean are reported to allow assessment of bias. The coefficient of variation reported (in column `cv`) is based on the median calculated from the bootstraps.

Value

a `data.frame` of summary statistics

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

Description

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

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

Systematic_variance_1 *Simulation of encounter rate variance*

Description

systematic_var_1 consists of simulated line transect data with large differences in transect length. In systematic_var_2 that transect length gradient is coupled with a strong animal gradient; exaggerating encounter rate variance between transects.

Format

data.frame with 253 observations (systematic_var_1) or 256 observations (systematic_var_2) of 7 variables: Region.Label stratum label (default) Area stratum area (0.5 km^2) Sample.Label transect identifier Effort transect length (km) object object ID distance perpendicular distance (m) Study.Area name of study area

Details

True population size is 1000 objects in the study area of size 0.5 km^2; such that true density is 2000 objects per km.

References

Fewster, R.M., S.T. Buckland, K.P. Burnham, D.L. Borchers, P.E. Jupp, J.L. Laake and L. Thomas. (2009) Estimating the encounter rate variance in distance sampling. Biometrics 65 (1): 225–236. doi:10.1111/j.15410420.2008.01018.x

unflatten *Unflatten flatfile data.frames*

Description

Sometimes data is provided in the flatfile format, but we really want it in mrds format (that is, as distance data, observation table, sample table and region table format). This function undoes the flattening, assuming that the data have the correct columns.

Usage

```
unflatten(data)
```

Arguments

data	data in flatfile format (a data.frame)

Value

list of four data.frames: distance data, observation table, sample table, region table.

Author(s)

David L Miller

`unimak` *Simulated line transect survey data with covariates*

Description

Simulated line transect survey. Only eight line transects, detection function is half-normal.

Format

A `data.frame` with 60 rows and 9 variables

- `Region.Label` strata names (single stratum)
- `Area` size of study area (mi^2)
- `Sample.Label` transect ID
- `Effort` transect length (mi)
- `object` object ID
- `distance` perpendicular distance (km)
- `MSTDO` time since medication taken by observer (min)
- `Hour` time of day of sighting (hour)
- `Study.Area` name of study area

Note

`Hour` is covariate that has no effect on detection function, while `MSTDO` does affect the detection function. Examine the ability of model selection to choose the correct model.

Source

Simulated data, from the distance sampling introductory course, Centre for Research into Ecological & Environmental Modelling, University of St Andrews.

units_table *Generate table of unit conversions*

Description

Returns a table of conversions between the units used in Distance for Windows. This is extracted from the `DistIni.mdb` default database.

Usage

```
units_table()
```

Author(s)

David L Miller

wren *Steve Buckland's winter wren surveys*

Description

Observations of winter wren (*Troglodytes troglodytes L.*) collected by Steve Buckland in woodland/parkland at Montrave Estate near Leven, Fife, Scotland.

Details

Four different surveys were carried out:

- `wren_5min` 5-minute point count
- `wren_snapshot` snapshot method
- `wren_cuecount` cue count
- `wren_lt` line transect survey

Note

`wren_5min`: 134 observations of 8 variables

- `Region.Label` stratum name (single stratum)
- `Area` size (ha) of Montrave study area
- `Sample.Label` point label
- `Effort` Number of visits to point
- `object` Object ID
- `distance` radial distance (m)

- direction direction of detection from point
- Study.Area Montrave Estate

wren_snapshot: 119 observations of 7 variables

- Region.Label stratum name (single stratum)
- Area size (ha) of Montrave study area
- Sample.Label point label
- Effort Number of visits to point
- object Object ID
- distance radial distance (m)
- Study.Area Montrave Estate

wren_cuecount: 774 observations of 9 variables

- Region.Label stratum name (single stratum)
- Area size (ha) of Montrave study area
- Sample.Label point label
- Cue.rate Production rate (per min) of cues
- Cue.rate.SE SE of cue production rate
- object Object ID
- distance radial distance (m)
- Search.time Time (min) listening for cues
- Study.Area Montrave Estate

wren_lt: 156 observations of 8 variables

- Region.Label stratum name (single stratum)
- Area size (ha) of Montrave study area
- Sample.Label transect label
- Effort transect length (km)
- object Object ID
- distance perpendicular distance (m)
- Study.Area Montrave Estate

Source

Steve Buckland

References

Buckland, S. T. (2006) Point-transect surveys for songbirds: robust methodologies. The Auk 123 (2): 345–357.

Index

∗ **Models**
  Distance-package, 3
∗ **Statistical**
  Distance-package, 3
∗ **datasets**
  amakihi, 6
  capercaillie, 11
  ClusterExercise, 13
  CueCountingExample, 16
  ducknest, 30
  DuikerCameraTraps, 31
  ETP_Dolphin, 32
  golftees, 36
  LTExercise, 38
  minke, 40
  PTExercise, 46
  Savannah_sparrow_1980, 49
  sikadeer, 50
  Stratify_example, 51
  Systematic_variance_1, 54
  unimak, 55
  wren, 56
∗ **utility**
  ds.gof, 29
  print.summary.dsmodel, 46
  summary.dsmodel, 53

activity::fitact, *39*
add_df_covar_line, 4, *28*, *42*
AIC.ds, *28*
AIC.dsmodel, 5
amakihi, 6
amakihi_units *(*amakihi*)*, 6

bootdht, 7, *10, 11*, *39*, *52*
bootdht_Dhat_summarize, *9*, 10, *11*
bootdht_Nhat_summarize, *7*, *9, 10*, 11

capercaillie, 11
capercaillie_units *(*capercaillie*)*, 11
check.mono, *27*
checkdata, 12
chi2_select *(*QAIC*)*, 47
ClusterExercise, 13
ClusterExercise_units
        *(*ClusterExercise*)*, 13
convert_units, 13
create.bins, 14
create_bins, *14*, 15
CueCountingExample, 16
CueCountingExample_units
        *(*CueCountingExample*)*, 16

ddf, *17, 18*, *43*
ddf.gof, *30*, *35*
dht, *7*, *26*, *41*, *53*
dht.se, *25*
dht2, *10, 11*, 17, *25, 26*, *33*, *45*
Distance *(*Distance-package*)*, 3
Distance-package, 3
ds, *7*, *17, 18*, 22, *45*
ds.gof, *28*, 29
ducknest, 30
ducknest_units *(*ducknest*)*, 30
ducknests_units *(*ducknest*)*, 30
DuikerCameraTraps, 31
DuikerCameraTraps_units
        *(*DuikerCameraTraps*)*, 31
dummy_ddf, 31

ETP_Dolphin, 32
ETP_Dolphin_units *(*ETP_Dolphin*)*, 32

flatfile, *7*, *17*, *23*, *28*, 33, *40*, *54*

gof_ds, *29*, 35
golftees, 36
golftees_units *(*golftees*)*, 36

hist, *4*

legend, *4*
lines, *4*
logLik.dsmodel, 37
LTExercise, 38
LTExercise_units *(*LTExercise*)*, 38

make_activity_fn, *8*, 39
mcds_dot_exe, *25*
minke, 40
mrds::predict, *44*
mrds_opt, *25, 26*

optim, *25*
optimx, *25*

p_dist_table, *28*, 41
plot.ds, *28*, *42*
plot.dsmodel, 42
predict.dsmodel, 43
predict.fake_ddf, 44
print.dht_result, 45
print.dsmodel, 45
print.summary.dsmodel, 46
PTExercise, 46
PTExercise_units *(*PTExercise*)*, 46

QAIC, 47
qqplot.ddf, *30*, *35*
quantile, *10, 11*

read.csv, *33*
readxl::read_xls, *33*

Savannah_sparrow_1980, 49
Savannah_sparrow_1980_units
*(*Savannah_sparrow_1980*)*, 49
Savannah_sparrow_1981
*(*Savannah_sparrow_1980*)*, 49
Savannah_sparrow_1981_units
*(*Savannah_sparrow_1980*)*, 49
set.seed, *9*
sikadeer, 50
sikadeer_units *(*sikadeer*)*, 50
Stratify_example, 51
Stratify_example_units
*(*Stratify_example*)*, 51
summarize_ds_models, 51
summary, *45*
summary.dht_bootstrap, *9*, 52
summary.ds, *46*, *53*
summary.dsmodel, 53
Systematic_variance_1, 54
Systematic_variance_1_units
*(*Systematic_variance_1*)*, 54
Systematic_variance_2
*(*Systematic_variance_1*)*, 54
Systematic_variance_2_units
*(*Systematic_variance_1*)*, 54

unflatten, 54
unimak, 55
unimak_units *(*unimak*)*, 55
units_table, *14*, 56

varn, *18*, *20*

wren, 56
wren_5min *(*wren*)*, 56
wren_5min_units *(*wren*)*, 56
wren_cuecount *(*wren*)*, 56
wren_cuecount_units *(*wren*)*, 56
wren_lt *(*wren*)*, 56
wren_lt_units *(*wren*)*, 56
wren_snapshot *(*wren*)*, 56
wren_snapshot_units *(*wren*)*, 56