Package 'Rdistance'

April 10, 2025

Type Package

Title Density and Abundance from Distance-Sampling Surveys

Version 4.0.5

Date 2025-04-10

Maintainer Trent McDonald <trent@mcdonalddatasciences.com>

Description Distance-sampling (<doi:10.1007/978-3-319-19219-2>) estimates density and abundance of survey targets (e.g., animals) when detection probability declines with distance.
Distance-sampling is popular in ecology,
especially when survey targets are observed from aerial platforms (e.g., airplane or drone), surface vessels (e.g., boat or truck), or along walking transects.
Distance-sampling includes line-transect studies that measure observation distances as the closest approach of the sample route (transect) to the target (i.e., perpendicular off-transect distance), and point-transect studies that measure observation distances from stationary observers to the target (i.e., radial distance).
The routines included here fit smooth (parametric) curves to histograms of observation distances
and use those functions to compute effective sampling distances, density of targets in the surveyed area, and abundance
of targets in a surrounding study area. Curve shapes include the half-normal, hazard rate, and negative exponential functions.
Physical measurement units are required and used throughout to ensure density is reported correctly. The help files
are extensive and have been vetted by multiple authors.

License GNU General Public License

URL https://github.com/tmcd82070/Rdistance/wiki

BugReports https://github.com/tmcd82070/Rdistance/issues

Suggests testthat (>= 3.0.0),

Depends R (>= 4.1.0), units

Imports graphics, stats, utils, crayon, withr, tidyr, dplyr, progress, tibble, tidyselect

RoxygenNote 7.3.2

Encoding UTF-8

Config/testthat/edition 3

NeedsCompilation no

Author Trent McDonald [cre, aut],
Jason Carlisle [aut],
Aidan McDonald [aut] (point transect methods),
Ryan Nielson [ctb] (smoothed likelihood),
Ben Augustine [ctb] (maximization method),
James Griswald [ctb] (maximization method),
Patrick McKann [ctb] (maximization method),
Lacey Jeroue [ctb] (vignettes),
Hoffman Abigail [ctb] (vignettes),
Kleinsausser Michael [ctb] (vignettes),
Joel Reynolds [ctb] (Gamma likelihood),
Pham Quang [ctb] (Gamma likelihood),
Earl Becker [ctb] (Gamma likelihood),
Aaron Christ [ctb] (Gamma likelihood),
Brook Russelland [ctb] (Gamma likelihood),
Stefan Emmons [ctb] (Automated tests),
Will McDonald [ctb] (Automated tests),
Reid Olson [ctb] (Automated tests and bug fixes)

Repository CRAN

Date/Publication 2025-04-10 17:20:02 UTC

Contents

`Rdistance-package` *Rdistance - Distance Sampling Analyses for Abundance Estimation*

Description

`Rdistance` contains functions and associated routines to analyze distance-sampling data collected on point or line transects. Some of `Rdistance`'s features include:

- Accommodation of both point and line transect analyses in one routine (`dfuncEstim`).
- Regression-like formula for inclusion of distance function covariates (`dfuncEstim`).
- Automatic bootstrap confidence intervals (`abundEstim`).
- Availability of both study-area and site-level abundance estimates (`help("predict.dfunc")`).
- Classical, parametric distance functions (`halfnorm.like`, `hazrate.like`, `negexp.like`), and expansion functions (`cosine.expansion`, `hermite.expansion`, `simple.expansion`).
- Automated distance function fits and selection `autoDistSamp`.
- `print`, `plot`, `predict`, `coef`, and `summary` methods for distance function objects and abundance classes.

Background

Distance-sampling is a popular method for abundance estimation in ecology. Line transect surveys are conducted by traversing randomly placed transects in a study area with the objective of sighting animals and estimating density or abundance. Data collected during line transect surveys consists of sighting records for *targets*, usually either individuals or groups of individuals. Among the collected data, off-transect distances are recorded or computed from other information (see `perpDists`). Off-transect distances are the perpendicular distances from the transect to the location of the initial sighting cue. When groups are the target, the number of individuals in the group is recorded.

Point transect surveys are similar except that observers stop one or more times along the transect to observe targets. This is a popular method for avian surveys where detections are often auditory cues, but is also appropriate when automated detectors are placed along a route. Point transect surveys collect distances from the observer to the target and are sometimes called *radial* distances.

A fundamental characteristic of both line and point-based distance sampling analyses is that probability of detecting a target declines as off-transect or radial distances increase. Targets far from the observer are usually harder to detect than closer targets. In most classical line transect studies, targets on the transect (off-transect distance = 0) are assume to be sighted with 100% probability. This assumption allows estimation of the proportion of targets missed during the survey, and thus it is possible to adjust the actual number of sighted targets for the proportion of targets missed. Some studies utilize two observers searching the same areas to estimate the proportion of individuals missed and thereby eliminating the assumption that all individuals on the line have been observed.

Relationship to other software

A detailed comparison of Rdistance to other options for distance sampling analysis (e.g., Program DISTANCE, R package Distance, and R package unmarked) is forthcoming. While some of the functionality in Rdistance is not unique, our aim is to provide an easy-to-use, rigorous, and flexible analysis option for distance-sampling data. We understand that beginning users often need software that is both easy to use and easy to understand, and that advanced users often require greater flexibility and customization. Our aim is to meet the demands of both user groups. Rdistance is under active development, so please contact us with issues, feature requests, etc. through the package's GitHub website (https://github.com/tmcd82070/Rdistance).

Data sets

Rdistance contains four example data sets: two collected using line-transect methods (i.e., sparrowDetectionData and sparrowSiteData) and two collected using point-transect methods (i.e., thrasherDetectionData and thrasherSiteData).

Author(s)

Main author and maintainer: Trent McDonald <trent@mcdonalddatasciences.com>

Coauthors: Ryan Nielson, Jason Carlisle, and Aidan McDonald

Contributors: Ben Augustine, James Griswald, Joel Reynolds, Pham Quang, Earl Becker, Aaron Christ, Brook Russelland, Patrick McKann, Lacey Jeroue, Abigail Hoffman, Michael Kleinsasser, and Ried Olson

References

Buckland, S.T., Anderson, D.R., Burnham, K.P. and Laake, J.L. 1993. *Distance Sampling: Estimating Abundance of Biological Populations*. Chapman and Hall, London.

See Also

Useful links:

- https://github.com/tmcd82070/Rdistance/wiki
- Report bugs at https://github.com/tmcd82070/Rdistance/issues

---

abundEstim *abundEstim - Distance Sampling Abundance Estimates*

Description

Estimate abundance (or density) from an estimated detection function and supplemental information on observed group sizes, transect lengths, area surveyed, etc. Computes confidence intervals on abundance (or density) using a the bias corrected bootstrap method.

Usage

```
abundEstim(
  object,
  area = NULL,
  propUnitSurveyed = 1,
  ci = 0.95,
  R = 500,
  plot.bs = FALSE,
  showProgress = TRUE
)
```

Arguments

object	An Rdistance model frame or fitted distance function, normally produced by a call to dfuncEstim.
area	A scalar containing the total area of inference. Usually, this is study area size. If area is NULL (the default), area will be set to 1 square unit of the output units and density estimates will be produced. If area is not NULL, it must have measurement units assigned by the units package. The units on area must be convertible to squared output units. Units on area must be two-dimensional. For example, if output units are "foo", units on area must be convertible to "foo^2" by the units package. Units of "km^2", "cm^2", "ha", "m^2", "acre", "mi^2", and several others are acceptable.
propUnitSurveyed	A scalar or vector of real numbers between 0 and 1. The proportion of the default sampling unit that was surveyed. If both sides of line transects were observed, propUnitSurveyed = 1. If only a single side of line transects were observed, set propUnitSurveyed = 0.5. For point transects, this should be set to the proportion of each circle that was observed. Length must either be 1 or the total number of transects in x.
ci	A scalar indicating the confidence level of confidence intervals. Confidence intervals are computed using a bias corrected bootstrap method. If ci = NULL or ci == NA, confidence intervals are not computed.
R	The number of bootstrap iterations to conduct when ci is not NULL.
plot.bs	A logical scalar indicating whether to plot individual bootstrap iterations.
showProgress	A logical indicating whether to show a text-based progress bar during bootstrapping. Default is TRUE. It is handy to shut off the progress bar if running this within another function. Otherwise, it is handy to see progress of the bootstrap iterations.

Details

The abundance estimate for line-transect surveys (if no covariates are included in the detection function and both sides of the transect are observed) is

$$N = \frac{n(A)}{2(ESW)(L)}$$

where *n* is total number of sighted individuals (i.e., `sum(groupSizes(dfunc))`), *L* is the total length of surveyed transect (i.e., `sum(effort(dfunc))`), and *ESW* is effective strip width computed from the estimated distance function (i.e., `ESW(dfunc)`). If only one side of transects were observed, the "2" in the denominator is not present (or, replaced with a "1").

The abundance estimate for point transect surveys (if no covariates are included) is

$$N = \frac{n(A)}{\pi(ESR^2)(P)}$$

where *n* is total number of sighted individuals (i.e., `sum(groupSizes(dfunc))`), *P* is the total number of surveyed points (i.e., `sum(effort(dfunc))`), and *ESR* is effective search radius computed from the estimated distance function (i.e., `ESR(dfunc)`).

Setting `plot.bs=FALSE` and `showProgress=FALSE` suppresses all intermediate output.

Estimation of site-specific density (e.g., on every transect) is accomplished by `predict(x, type = "density")`, which returns a tibble containing density and abundance on the area surveyed by every transect.

Value

An Rdistance 'abundance estimate' object, which is a list of class `c("abund", "dfunc")`, containing all the components of a "dfunc" object (see `dfuncEstim`), plus the following:

`estimates`	A tibble containing fitted coefficients in the distance function, density in the area(s) surveyed, abundance on the study area, the number of groups seen between w.lo and w.hi, the number of individuals seen between w.lo and w.hi, study area size, surveyed area, average group size, and average effective detection distance.
`B`	If confidence intervals were requested, a tibble containing all bootstrap values of coefficients, density, abundance, groups seen, individuals seen, study area size, surveyed area size, average group size, and average effective detection distance. The number of rows is always `R`, the requested number of bootstrap iterations. If an iteration fails, the corresponding row in `B` is `NA` (hence, use 'na.rm = TRUE' when computing summaries). Columns 1 through `length(coef(dfunc))` contain bootstrap realizations of the distance function's coefficients.
`ci`	Confidence level of the confidence intervals

Bootstrap Confidence Intervals

Rdistance's nested data frames (produced by `RdistDf`) contain all information required to estimate bootstrap CIs. To compute bootstrap CIs, Rdistance resamples, with replacement, the rows of the `$data` component contained in Rdistance fitted models. Rdistance assumes each row of `$data` contains one information on on transect. The `$data` component also contains information on which observations go into the detection functions, which should be counted as detected targets, and which count toward transect length. After resampling rows of `$data`, Rdistance refits the distance function using non-missing distances, recomputes the detected number of targets using non-missing group sizes on transects with non-missing length, and re-computes total transect length from transects with non-missing lengths. By default, `R` = 500 bootstrap iterations are performed, after which bias corrected confidence intervals are computed (Manly, 1997, section 3.4).

The distance function is not re-selected during bootstrap resampling. The model of the input object is re-fitted every iteration.

During bootstrap iterations, the distance function can fail. An iteration can fail for a two reasons: (1) no detections on the iteration, and (2) a bad configuration of distances that push the distance function's parameters to their limits. When an iteration fails, Rdistance skips the iteration and effectively ignores the failed iterations. If the proportion of failed iterations is small (less than 20 is probably valid and no warning is issued. If the proportion of non-convergent iterations is not small (exceeds 20 The warning can be modified by re-setting the `Rdistance_maxBSFailPropForWarning` option. Setting `options(Rdistance_masBSFailPropForWarning = 1.0)` will turn off the warning. Setting `options(Rdistance_masBSFailPropForWarning = 0.0)` will warn if any iteration failed. Results (density and effective sampling distance) from all successful iterations are contained in the non-NA rows of data frame 'B' in the output object.

Missing Transect Lengths

Transect lengths can be missing in the RdistDf object. Missing length transects are equivalent to 0 [m] transects and do not count toward total surveyed units nor to group sizes on these transects count toward total detected individuals. Use NA-length transects to include their associated distances when estimating the distance function, but not when estimating abundance. For example, this allows estimation of abundance on one study area using off-transect distances from another. This allows sightability to be estimated using two or more similar targets (e.g., two similar species), but abundance to be estimated separate for each target type. Include NA-length transects by including the "extra" distance observations in the detection data frame, with valid site IDs, but set the length of those site IDs to NA in the site data frame.

Point Transect Lengths

Point transects do not have a physical measurement for length. The "length" of point transects is the number of points on the transect. Point transects can contain only one point. Rdistance treats transects of points as independent and bootstrap resamples them to estimate variance. The number of points on each point transect must exist in the RdistDf and cannot have physical measurement units (it is a count, not a distance).

References

Manly, B.F.J. (1997) *Randomization, bootstrap, and Monte-Carlo methods in biology*, London: Chapman and Hall.

Buckland, S.T., D.R. Anderson, K.P. Burnham, J.L. Laake, D.L. Borchers, and L. Thomas. (2001) *Introduction to distance sampling: estimating abundance of biological populations*. Oxford University Press, Oxford, UK.

See Also

`dfuncEstim`, `autoDistSamp`, `predict.dfunc` with 'type = "density"'.

Examples

```
# Load example sparrow data (line transect survey type)
# sparrowDf <- RdistDf(sparrowSiteData, sparrowDetectionData)
```

```
data(sparrowDf)

# Fit half-normal detection function
dfunc <- sparrowDf |>
  dfuncEstim(formula=dist ~ groupsize(groupsize)
           , likelihood="halfnorm"
           , w.hi=units::set_units(150, "m")
  )

# Estimate abundance - Convenient for programming
abundDf <- estimateN(dfunc
                     , area = units::set_units(4105, "km^2")
           )

# Same - Nicer output
# Set ci=0.95 (or another value) to estimate bootstrap CI's on ESW, density, and abundance.
fit <- abundEstim(dfunc
                , area = units::set_units(4105, "km^2")
                , ci = NULL
                )
```

`AIC.dfunc` *AIC.dfunc - AIC-related fit statistics for detection functions*

Description

Computes AICc, AIC, or BIC for estimated distance functions.

Usage

```
## S3 method for class 'dfunc'
AIC(object, ..., criterion = "AICc")
```

Arguments

`object`	An Rdistance model frame or fitted distance function, normally produced by a call to `dfuncEstim`.
`...`	Included for compatibility with generic `predict` methods.
`criterion`	String specifying the criterion to compute. Either "AICc", "AIC", or "BIC".

Details

Regular Akaike's information criterion (https://en.wikipedia.org/wiki/Akaike_information_criterion) (AIC) is

$$AIC = LL + 2p,$$

where LL is the maximized value of the log likelihood (the minimized value of the negative log likelihood) and p is the number of coefficients estimated in the detection function. For `dfunc` objects, AIC = `obj$loglik + 2*length(coef(obj))`.

A correction for small sample size, AIC_c, is

$$AIC_c = LL + 2p + \frac{2p(p+1)}{n-p-1},$$

where n is sample size or number of detected groups for distance analyses. By default, this function computes AIC_c. AIC_c converges quickly to AIC as n increases.

The Bayesian Information Criterion (BIC) is

$$BIC = LL + log(n)p,$$

.

Value

A scalar, the requested fit statistic for `object`.

References

Burnham, K. P., and D. R. Anderson, 2002. *Model selection and multi-model inference: A practical information-theoretic approach, Second ed.* Springer-Verlag. ISBN 0-387-95364-7.

McQuarrie, A. D. R., and Tsai, C.-L., 1998. *Regression and time series model selection.* World Scientific. ISBN 981023242X

See Also

`coef`, `dfuncEstim`

Examples

```
data(sparrowDf)
dfunc <- sparrowDf |> dfuncEstim(dist~1)

# Fit statistics
AIC(dfunc)  # AICc
AIC(dfunc, criterion="AIC")  # AIC
AIC(dfunc, criterion="BIC")  # BIC
```

`autoDistSamp` *autoDistSamp - Automated classical distance analysis*

Description

Perform automated likelihood, expansion, and series selection for a classic distance sampling analysis. Estimate abundance using the best fitting likelihood, expansion, and series.

Usage

```
autoDistSamp(
  data,
  formula,
  likelihoods = c("halfnorm", "hazrate", "negexp"),
  w.lo = units::set_units(0, "m"),
  w.hi = NULL,
  expansions = 0:3,
  series = c("cosine"),
  x.scl = w.lo,
  g.x.scl = 1,
  warn = TRUE,
  outputUnits = NULL,
  area = NULL,
  propUnitSurveyed = 1,
  ci = 0.95,
  R = 500,
  plot.bs = FALSE,
  showProgress = TRUE,
  plot = TRUE,
  criterion = "AICc"
)
```

Arguments

data	An RdistDf data frame. RdistDf data frames contain one line per transect and a list-based column. The list-based column contains a data frame with detection information. The detection information data frame on each row contains (at least) distances and group sizes of all targets detected on the transect. Function RdistDf creates RdistDf data frames from separate transect and detection data frames. is.RdistDf checks whether data frames are RdistDf's.
formula	A standard formula object. For example, dist ~ 1, dist ~ covar1 + covar2). The left-hand side (before ~) is the name of the vector containing off-transect or radial detection distances. The right-hand side contains the names of covariate vectors to fit in the detection function, and potentially group sizes. Covariates can be either detection level or transect level and can appear in data or exist in the global working environment. Regular R scoping rules apply.
likelihoods	String vector specifying the likelihoods to fit. See 'likelihood' parameter of dfuncEstim.
w.lo	Lower or left-truncation limit of the distances in distance data. This is the minimum possible off-transect distance. Default is 0. If w.lo is greater than 0, it must be assigned measurement units using units(w.lo) <- "<units>" or w.lo <- units::set_units(w.lo, "<units>"). See examples in the help for set_units.
w.hi	Upper or right-truncation limit of the distances in dist. This is the maximum off-transect distance that could be observed. If unspecified (i.e., NULL), right-truncation is set to the maximum of the observed distances. If w.hi is

specified, it must have associated measurement units. Assign measurement units using `units(w.hi) <- "<units>"` or `w.hi <- units::set_units(w.hi, "<units>")`. See examples in the help for `set_units`.

`expansions` A scalar specifying the number of terms in `series` to compute. Depending on the series, this could be 0 through 5. The default of 0 equates to no expansion terms of any type. No expansion terms are allowed (i.e., `expansions` is forced to 0) if covariates are present in the detection function (i.e., right-hand side of `formula` includes something other than 1).

`series` If `expansions` > 0, this string specifies the type of expansion to use. Valid values at present are 'simple', 'hermite', and 'cosine'.

`x.scl` The x coordinate (a distance) at which the detection function will be scaled. `g.x.scl` can be a distance or the string "max". When `x.scl` is specified (i.e., not 0 or "max"), it must have measurement units assigned using either `library(units);units(x.scl) <- '<units>'` or `x.scl <- units::set_units(x.scl, <units>)`. See `units::valid_udunits()` for valid symbolic units.

`g.x.scl` Height of the distance function at coordinate `x`. The distance function will be scaled so that g(`x.scl`) = `g.x.scl`. If `g.x.scl` is not a data frame, it must be a numeric value (vector of length 1) between 0 and 1.

`warn` A logical scalar specifying whether to issue an R warning if the estimation did not converge or if one or more parameter estimates are at their boundaries. For estimation, `warn` should generally be left at its default value of `TRUE`. When computing bootstrap confidence intervals, setting `warn` = `FALSE` turns off annoying warnings when an iteration does not converge. Regardless of `warn`, after completion all messages about convergence and boundary conditions are printed by `print.dfunc`, `print.abund`, and `plot.dfunc`.

`outputUnits` A string specifying the symbolic measurement units for results. Valid units are listed in `units::valid_udunits()`. The strings for common distance symbolic units are: "m" - meters, "ft" - feet, "cm" - centimeters, "mm" - millimeters, "mi" - miles, "nmile" - nautical miles ("nm" is nano meters), "in" - inches, "yd" - yards, "km" - kilometers, "fathom" - fathoms, "chains" - chains, and "furlong" - furlongs. If `outputUnits` is unspecified (NULL), output units will be the same as those on distances in `data`.

`area` A scalar containing the total area of inference. Usually, this is study area size. If `area` is NULL (the default), `area` will be set to 1 square unit of the output units and density estimates will be produced. If `area` is not NULL, it must have measurement units assigned by the `units` package. The units on `area` must be convertible to squared output units. Units on `area` must be two-dimensional. For example, if output units are "foo", units on area must be convertible to "foo^2" by the `units` package. Units of "km^2", "cm^2", "ha", "m^2", "acre", "mi^2", and several others are acceptable.

`propUnitSurveyed`

A scalar or vector of real numbers between 0 and 1. The proportion of the default sampling unit that was surveyed. If both sides of line transects were observed, `propUnitSurveyed` = 1. If only a single side of line transects were observed, set `propUnitSurveyed` = 0.5. For point transects, this should be set to the proportion of each circle that was observed. Length must either be 1 or the total number of transects in `x`.

ci	A scalar indicating the confidence level of confidence intervals. Confidence intervals are computed using a bias corrected bootstrap method. If ci = NULL or ci == NA, confidence intervals are not computed.
R	The number of bootstrap iterations to conduct when ci is not NULL.
plot.bs	A logical scalar indicating whether to plot individual bootstrap iterations.
showProgress	A logical indicating whether to show a text-based progress bar during bootstrapping. Default is TRUE. It is handy to shut off the progress bar if running this within another function. Otherwise, it is handy to see progress of the bootstrap iterations.
plot	Logical scalar specifying whether to plot models during model selection. If TRUE, a histogram with fitted distance function is plotted for every model. The function pauses between each plot and prompts the user for whether they want to continue. To suppress user prompts, set plot = FALSE.
criterion	A string specifying the criterion to use when assessing model fit. The best fitting model, as defined by this routine, has the lowest value of this criterion. This must be one of "AICc" (the default), "AIC", or "BIC". See AIC.dfunc for formulas.

Details

During distance function selection, all combinations of likelihoods, series, and number of expansions is fitted. For example, if likelihoods has 3 elements, series has 2 elements, and expansions has 4 elements, this routine fits a total of 3 (likelihoods) * 2 (series) * 4 (expansions) = 24 models. Default parameters fit 9 detection functions, i.e., all combinations of "halfnorm", "hazrate", and "negexp" likelihoods and 0 through 3 expansions. Other combinations are specified through values of likelihoods, series, and expansions.

Suppress all intermediate output using plot.bs=FALSE, showProgress=FALSE, and plot=FALSE.

The returned abundance estimate object contains an additional component, the fitting table (a list of models fitted and criterion values) in component $fitTable.

Value

An Rdistance 'abundance estimate' object, which is a list of class c("abund", "dfunc"), containing all the components of a "dfunc" object (see dfuncEstim), plus the following:

estimates	A tibble containing fitted coefficients in the distance function, density in the area(s) surveyed, abundance on the study area, the number of groups seen between w.lo and w.hi, the number of individuals seen between w.lo and w.hi, study area size, surveyed area, average group size, and average effective detection distance.
B	If confidence intervals were requested, a tibble containing all bootstrap values of coefficients, density, abundance, groups seen, individuals seen, study area size, surveyed area size, average group size, and average effective detection distance. The number of rows is always R, the requested number of bootstrap iterations. If an iteration fails, the corresponding row in B is NA (hence, use 'na.rm = TRUE' when computing summaries). Columns 1 through length(coef(dfunc)) contain bootstrap realizations of the distance function's coefficients.
ci	Confidence level of the confidence intervals

See Also

dfuncEstim, abundEstim.

Examples

```
# Load example sparrow data (line transect survey type)
data(sparrowDf)

autoDistSamp(data = sparrowDf
           , formula = dist ~ groupsize(groupsize)
           , likelihoods = c("halfnorm","negexp")
           , expansions = 0
           , plot = FALSE
           , ci = NULL
           , area = units::set_units(1, "hectare")
)

## Not run:
autoDistSamp(data = sparrowDf
    , formula = dist ~ 1 + groupsize(groupsize)
    , ci = 0.95
    , area = units::set_units(1, "hectare")
)

## End(Not run)
```

---

bcCI *bcCI - Bias corrected bootstraps*

Description

Calculate bias-corrected confidence intervals for bootstrap data using methods in Manly textbook.

Usage

```
bcCI(x.bs, x, ci = 0.95)
```

Arguments

x.bs	A vector of bootstrap estimates of some quantity.
x	A scalar of the original estimate of the quantity.
ci	A scalar of the desired confidence interval coverage.

Value

A named vector containing the lower and upper endpoints of the bias-corrected bootstrap confidence interval.

checkNEvalPts *checkNEvalPts - Check number of numeric integration intervals*

Description

Check that number of integration intervals is odd and sufficiently large.

Usage

```
checkNEvalPts(nEvalPts)
```

Arguments

nEvalPts	An integer to check.

Value

The first element of nEvalPts is returned if it is acceptable. If nEvalPts is not acceptable, an error is thrown.

checkUnits *checkUnits - Check for the presence of units*

Description

Check for the presence of physical measurement units on key columns of an RdistDf data frame.

Usage

```
checkUnits(ml)
```

Arguments

ml	An Rdistance model list produced by parseModel containing a list of parameters for the distance model.

Value

The input ml list, with units of various quantities converted to common units. If a check fails, for example, a quantity does not have units, an error is thrown.

coef.dfunc *coef.dfunc - Coefficients of an estimated detection function*

Description

Extract distance model coefficients from an estimated detection function object.

Usage

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

Arguments

object	An Rdistance model frame or fitted distance function, normally produced by a call to dfuncEstim.
...	Ignored

Value

The estimated coefficient vector for the detection function. Length and interpretation of values vary depending on the form of the detection function and expansion terms.

See Also

AIC, dfuncEstim

Examples

```
data(sparrowDfuncObserver) # pre-estimated dfunc

# Same as sparrowDfuncObserver$par
coef(sparrowDfuncObserver)

## Not run:
data(sparrowDf)
dfunc <- sparrowDf |> dfuncEstim(dist~bare + observer,
                      w.hi=units::set_units(150, "m"))
coef(dfunc)

## End(Not run)
```

`colorize` *colorize - Add color to result if terminal accepts it*

Description

Add ANSI color to a string using the `crayon` package, if the R environment accepts color. This function is needed because of the need to determine whether output can be colorized. This determination is left up to `crayon::has_color()`.

In addition, for Rdistance results, we want to only colorize numbers, not the reporting units. Everything between the last set of square brackets (`[...]`) is NOT colorized.

Usage

```
colorize(STR, col = NULL, bg = NULL)
```

Arguments

`STR`	The string to colorize.
`col`	A string specifying the desired foreground color. This is passed straight to `crayon::style` and so must be recognized as one of the 8 base crayon colors. i.e., "black", "red", "green", "yellow", "blue", "magenta", "cyan", "white", and "silver" (silver = gray). By default, numbers are styled in "green".
`bg`	A string specifying the desired background color. Must be one of "bgBlack", "bgRed", "bgGreen", "bgYellow", "bgBlue" "bgMagenta", "bgCyan", or "bgWhite". By default, no background is applied.

Value

If color is not allowed in the terminal, the input string is returned unperturbed. If color is allowed, the input string is returned with color and background ANSI code surrounding the initial part of the string from character 1 to the character before the [in the last pair of [].

See Also

`crayon::style`

`cosine.expansion` *cosine.expansion - Cosine expansion terms*

Description

Computes the cosine expansion terms used to modify the shape of distance likelihood functions.

Usage

```
cosine.expansion(x, expansions)
```

Arguments

x	A numeric vector of distances at which to evaluate the expansion series. For distance analysis, x is of the proportion of a strip transect's half-width at which a group of individuals were sighted, i.e., d/w.
expansions	A scalar specifying the number of expansion terms to compute. Must be one of the integers 1, 2, 3, 4, or 5.

Details

There are, in general, several expansions that can be called cosine. The cosine expansion used here is:

- **First term**:
$$h_1(x) = \cos(2\pi x),$$
- **Second term**:
$$h_2(x) = \cos(3\pi x),$$
- **Third term**:
$$h_3(x) = \cos(4\pi x),$$
- **Fourth term**:
$$h_4(x) = \cos(5\pi x),$$
- **Fifth term**:
$$h_5(x) = \cos(6\pi x),$$

The maximum number of expansion terms computed is 5.

Value

A matrix of size `length(x)` X `expansions`. The columns of this matrix are the cosine expansions of `x`. Column 1 is the first expansion term of `x`, column 2 is the second expansion term of `x`, and so on up to `expansions`.

See Also

`dfuncEstim`, `hermite.expansion`, `simple.expansion`, and the discussion of user defined likelihoods in `dfuncEstim`.

Examples

```
x <- seq(0, 1, length = 200)
cos.expn <- cosine.expansion(x, 5)
plot(range(x), range(cos.expn), type="n")
matlines(x, cos.expn, col=rainbow(5), lty = 1)
```

dE.multi *dE.multi - Estimate multiple-observer line-transect distance functions*

Description

Fits a detection function to off-transect distances collected by multiple observers.

Usage

```
dE.multi(
  data,
  formula,
  likelihood = "halfnorm",
  w.lo = units::set_units(0, "m"),
  w.hi = NULL,
  expansions = 0,
  series = "cosine",
  x.scl = units::set_units(0, "m"),
  g.x.scl = 1,
  warn = TRUE,
  outputUnits = NULL
)
```

Arguments

data	An RdistDf data frame. RdistDf data frames contain one line per transect and a list-based column. The list-based column contains a data frame with detection information. The detection information data frame on each row contains (at least) distances and group sizes of all targets detected on the transect. Function RdistDf creates RdistDf data frames from separate transect and detection data frames. is.RdistDf checks whether data frames are RdistDf's.
formula	A standard formula object. For example, dist ~ 1, dist ~ covar1 + covar2). The left-hand side (before ~) is the name of the vector containing off-transect or radial detection distances. The right-hand side contains the names of covariate vectors to fit in the detection function, and potentially group sizes. Covariates can be either detection level or transect level and can appear in data or exist in the global working environment. Regular R scoping rules apply.
likelihood	String specifying the likelihood to fit. Built-in likelihoods at present are "halfnorm", "hazrate", and "negexp".
w.lo	Lower or left-truncation limit of the distances in distance data. This is the minimum possible off-transect distance. Default is 0. If w.lo is greater than 0, it must be assigned measurement units using units(w.lo) <- "<units>" or w.lo <- units::set_units(w.lo, "<units>"). See examples in the help for set_units.

w.hi — Upper or right-truncation limit of the distances in dist. This is the maximum off-transect distance that could be observed. If unspecified (i.e., NULL), right-truncation is set to the maximum of the observed distances. If w.hi is specified, it must have associated measurement units. Assign measurement units using units(w.hi) <- "<units>" or w.hi <- units::set_units(w.hi, "<units>"). See examples in the help for set_units.

expansions — A scalar specifying the number of terms in series to compute. Depending on the series, this could be 0 through 5. The default of 0 equates to no expansion terms of any type. No expansion terms are allowed (i.e., expansions is forced to 0) if covariates are present in the detection function (i.e., right-hand side of formula includes something other than 1).

series — If expansions > 0, this string specifies the type of expansion to use. Valid values at present are 'simple', 'hermite', and 'cosine'.

x.scl — The x coordinate (a distance) at which the detection function will be scaled. g.x.scl can be a distance or the string "max". When x.scl is specified (i.e., not 0 or "max"), it must have measurement units assigned using either library(units);units(x.scl) <- '<units>' or x.scl <- units::set_units(x.scl, <units>). See units::valid_udunits() for valid symbolic units.

g.x.scl — Height of the distance function at coordinate x. The distance function will be scaled so that g(x.scl) = g.x.scl. If g.x.scl is not a data frame, it must be a numeric value (vector of length 1) between 0 and 1.

warn — A logical scalar specifying whether to issue an R warning if the estimation did not converge or if one or more parameter estimates are at their boundaries. For estimation, warn should generally be left at its default value of TRUE. When computing bootstrap confidence intervals, setting warn = FALSE turns off annoying warnings when an iteration does not converge. Regardless of warn, after completion all messages about convergence and boundary conditions are printed by print.dfunc, print.abund, and plot.dfunc.

outputUnits — A string specifying the symbolic measurement units for results. Valid units are listed in units::valid_udunits(). The strings for common distance symbolic units are: "m" - meters, "ft" - feet, "cm" - centimeters, "mm" - millimeters, "mi" - miles, "nmile" - nautical miles ("nm" is nano meters), "in" - inches, "yd" - yards, "km" - kilometers, "fathom" - fathoms, "chains" - chains, and "furlong" - furlongs. If outputUnits is unspecified (NULL), output units will be the same as those on distances in data.

Value

An object of class 'dfunc'. Objects of class 'dfunc' are lists containing the following components:

par — The vector of estimated parameter values. Length of this vector for built-in likelihoods is one (for the function's parameter) plus the number of expansion terms plus one if the likelihood is 'hazrate' (which has two parameters).

varcovar — The variance-covariance matrix for coefficients of the distance function, estimated by the inverse of the fit's Hessian evaluated at the estimates. Rdistance estimates the Hessian as the second derivative of the log likelihood surface at the

	final estimates, where second derivatives are estimated by numeric differentiation (see secondDeriv. There is no guarantee this matrix is positive-definite and should be viewed with caution. Error estimates derived from bootstrapping are generally more reliable. I.e., re-compute coefficient confidence intervals using the bootstrap values in component $B of an abundance object.
loglik	The maximized value of the log likelihood.
convergence	The convergence code. This code is returned by optim or nlminb. Values other than 0 indicate suspect convergence.
likelihood	The name of the likelihood. This is the value of the argument likelihood.
w.lo	Left-truncation value used during the fit.
w.hi	Right-truncation value used during the fit.
mf	A modelframe of detections within the strip or circle used in the fit. Column 'dist' contains the observed distances. Column 'offset(...)' contains group sizes associated with the values of 'dist'. Group sizes are only used in abundEstim. This model frame contains only non-missing distances between w.lo and w.hi.
model.frame	A model.frame object containing observed distances (the 'response'), covariates specified in the formula, and group sizes if they were specified. If specified, the name of the group size column is "offset(-variable-)", not "groupsize(-variable-)", because internally it is easier to treat group sizes as an offset in the model. This component is a proper model.frame and contains both 'terms' and 'contrasts' attributes.
siteID.cols	A vector containing the transect ID column names in detectionData and siteData. Transect IDs can be a composite of two or more columns and hence this component can have length greater than 1.
expansions	The number of expansion terms used during estimation.
series	The type of expansion used during estimation.
call	The original call of this function.
call.x.scl	The *input* or user requested distance at which the distance function is scaled.
call.g.x.scl	The input value specifying the height of the distance function at a distance of call.x.scl.
call.observer	The value of input parameter observer. The input observer parameter is only applicable when g.x.scl is a data frame.
fit	The fitted object returned by optim. See documentation for optim.
factor.names	The names of any factors in formula.
pointSurvey	The input value of pointSurvey. This is TRUE if distances are radial from a point. FALSE if distances are perpendicular off-transect.
formula	The formula specified for the detection function.
control	A list containing values of the 'control' parameters set by RdistanceControls.
outputUnits	The measurement units used for output. All distance measurements are converted to these units internally.
x.scl	The *actual* distance at which the distance function is scaled to some value. i.e., this is the actual x at which g(x) = g.x.scl. Note that call.x.scl = x.scl unless call.x.scl == "max", in which case x.scl is the distance at which $g()$ is maximized.

`g.x.scl` The *actual* height of the distance function at a distance of `x.scl`. Note that `g.x.scl = call.g.x.scl` unless `call.g.x.scl` is a multiple observer data frame, in which case `g.x.scl` is the actual height of the distance function at `x.scl` computed from the multiple observer data frame.

`dE.single` *dE.single - Estimate single-observer line-transect distance function*

Description

Fits a detection function to off-transect distances collected by a single observer.

Usage

```
dE.single(
  data,
  formula,
  likelihood = "halfnorm",
  w.lo = units::set_units(0, "m"),
  w.hi = NULL,
  expansions = 0,
  series = "cosine",
  x.scl = w.lo,
  g.x.scl = 1,
  warn = TRUE,
  outputUnits = NULL
)
```

Arguments

`data`	An `RdistDf` data frame. `RdistDf` data frames contain one line per transect and a list-based column. The list-based column contains a data frame with detection information. The detection information data frame on each row contains (at least) distances and group sizes of all targets detected on the transect. Function `RdistDf` creates `RdistDf` data frames from separate transect and detection data frames. `is.RdistDf` checks whether data frames are `RdistDf`'s.
`formula`	A standard formula object. For example, `dist ~ 1`, `dist ~ covar1 + covar2`). The left-hand side (before ~) is the name of the vector containing off-transect or radial detection distances. The right-hand side contains the names of covariate vectors to fit in the detection function, and potentially group sizes. Covariates can be either detection level or transect level and can appear in `data` or exist in the global working environment. Regular R scoping rules apply.
`likelihood`	String specifying the likelihood to fit. Built-in likelihoods at present are "halfnorm", "hazrate", and "negexp".

w.lo	Lower or left-truncation limit of the distances in distance data. This is the minimum possible off-transect distance. Default is 0. If w.lo is greater than 0, it must be assigned measurement units using units(w.lo) <- "<units>" or w.lo <- units::set_units(w.lo, "<units>"). See examples in the help for set_units.
w.hi	Upper or right-truncation limit of the distances in dist. This is the maximum off-transect distance that could be observed. If unspecified (i.e., NULL), right-truncation is set to the maximum of the observed distances. If w.hi is specified, it must have associated measurement units. Assign measurement units using units(w.hi) <- "<units>" or w.hi <- units::set_units(w.hi, "<units>"). See examples in the help for set_units.
expansions	A scalar specifying the number of terms in series to compute. Depending on the series, this could be 0 through 5. The default of 0 equates to no expansion terms of any type. No expansion terms are allowed (i.e., expansions is forced to 0) if covariates are present in the detection function (i.e., right-hand side of formula includes something other than 1).
series	If expansions > 0, this string specifies the type of expansion to use. Valid values at present are 'simple', 'hermite', and 'cosine'.
x.scl	The x coordinate (a distance) at which the detection function will be scaled. g.x.scl can be a distance or the string "max". When x.scl is specified (i.e., not 0 or "max"), it must have measurement units assigned using either library(units);units(x.scl) <- '<units>' or x.scl <- units::set_units(x.scl, <units>). See units::valid_udunits() for valid symbolic units.
g.x.scl	Height of the distance function at coordinate x. The distance function will be scaled so that g(x.scl) = g.x.scl. If g.x.scl is not a data frame, it must be a numeric value (vector of length 1) between 0 and 1.
warn	A logical scalar specifying whether to issue an R warning if the estimation did not converge or if one or more parameter estimates are at their boundaries. For estimation, warn should generally be left at its default value of TRUE. When computing bootstrap confidence intervals, setting warn = FALSE turns off annoying warnings when an iteration does not converge. Regardless of warn, after completion all messages about convergence and boundary conditions are printed by print.dfunc, print.abund, and plot.dfunc.
outputUnits	A string specifying the symbolic measurement units for results. Valid units are listed in units::valid_udunits(). The strings for common distance symbolic units are: "m" - meters, "ft" - feet, "cm" - centimeters, "mm" - millimeters, "mi" - miles, "nmile" - nautical miles ("nm" is nano meters), "in" - inches, "yd" - yards, "km" - kilometers, "fathom" - fathoms, "chains" - chains, and "furlong" - furlongs. If outputUnits is unspecified (NULL), output units will be the same as those on distances in data.

Details

Optimization and estimation controls can be modified using options(). See RdistanceControls.

Value

An object of class 'dfunc'. Objects of class 'dfunc' are lists containing the following components:

par	The vector of estimated parameter values. Length of this vector for built-in likelihoods is one (for the function's parameter) plus the number of expansion terms plus one if the likelihood is 'hazrate' (which has two parameters).
varcovar	The variance-covariance matrix for coefficients of the distance function, estimated by the inverse of the fit's Hessian evaluated at the estimates. Rdistance estimates the Hessian as the second derivative of the log likelihood surface at the final estimates, where second derivatives are estimated by numeric differentiation (see secondDeriv. There is no guarantee this matrix is positive-definite and should be viewed with caution. Error estimates derived from bootstrapping are generally more reliable. I.e., re-compute coefficient confidence intervals using the bootstrap values in component $B of an abundance object.
loglik	The maximized value of the log likelihood.
convergence	The convergence code. This code is returned by optim or nlminb. Values other than 0 indicate suspect convergence.
likelihood	The name of the likelihood. This is the value of the argument likelihood.
w.lo	Left-truncation value used during the fit.
w.hi	Right-truncation value used during the fit.
mf	A modelframe of detections within the strip or circle used in the fit. Column 'dist' contains the observed distances. Column 'offset(...)' contains group sizes associated with the values of 'dist'. Group sizes are only used in abundEstim. This model frame contains only non-missing distances between w.lo and w.hi.
model.frame	A model.frame object containing observed distances (the 'response'), covariates specified in the formula, and group sizes if they were specified. If specified, the name of the group size column is "offset(-variable-)", not "groupsize(-variable-)", because internally it is easier to treat group sizes as an offset in the model. This component is a proper model.frame and contains both 'terms' and 'contrasts' attributes.
siteID.cols	A vector containing the transect ID column names in detectionData and siteData. Transect IDs can be a composite of two or more columns and hence this component can have length greater than 1.
expansions	The number of expansion terms used during estimation.
series	The type of expansion used during estimation.
call	The original call of this function.
call.x.scl	The *input* or user requested distance at which the distance function is scaled.
call.g.x.scl	The input value specifying the height of the distance function at a distance of call.x.scl.
call.observer	The value of input parameter observer. The input observer parameter is only applicable when g.x.scl is a data frame.
fit	The fitted object returned by optim. See documentation for optim.
factor.names	The names of any factors in formula.
pointSurvey	The input value of pointSurvey. This is TRUE if distances are radial from a point. FALSE if distances are perpendicular off-transect.
formula	The formula specified for the detection function.

control	A list containing values of the 'control' parameters set by RdistanceControls.
outputUnits	The measurement units used for output. All distance measurements are converted to these units internally.
x.scl	The *actual* distance at which the distance function is scaled to some value. i.e., this is the actual x at which g(x) = g.x.scl. Note that call.x.scl = x.scl unless call.x.scl == "max", in which case x.scl is the distance at which $g()$ is maximized.
g.x.scl	The *actual* height of the distance function at a distance of x.scl. Note that g.x.scl = call.g.x.scl unless call.g.x.scl is a multiple observer data frame, in which case g.x.scl is the actual height of the distance function at x.scl computed from the multiple observer data frame.

Group Sizes

To specify non-unity group sizes, use groupsize() on the RHS of formula. When group sizes are not all 1, they must appear in a column of the 'detections' list-column of data. For example, d ~ habitat + groupsize(number) specifies distances in column d, one covariate named habitat, and that column number contains the number of individuals associated with each detection. If group sizes are not specified, all group sizes are assumed to be 1.

Contrasts

Factor contrasts in Rdistance are specified the same way as in lm or glm. By default, Rdistance uses contrasts in getOption("contrasts"). To change contrasts, use a statement like options(contrasts = c(unordered = "contr.SAS",ordered = "contr.poly")). Or, to set contrasts for a specific factor in the input data frame, use contrasts(df$A) <- "contr.sum" or similar. See contrasts or the contrasts.arg of model.matrix.

Transect types

Rdistance accommodates two kinds of transects: continuous and point. Detections can occur at any point on continuous transects. Rdistance calls these 'line-transects' even though routes are not necessarily a straight line. On point transects, detections occur at a series of stops (points). Rdisance calls these point-transects. Transects are the basic sampling unit in both cases. Rdistance assumes each row of data contains information from one transect. See RdistDf for more details.

Measurement Units

As of Rdistance version 3.0.0, measurement units are require on all physical distances. Requiring units ensures that internal calculations and results (e.g., ESW and abundance) are correct and that output units are clear. Physical distances are required on off-transect distances, radial distances, truncation distances (w.lo, unless it is zero; and w.hi, unless it is NULL), scale locations (x.scl, unless it is zero), line-transect lengths, and study area size. All units are 1-dimensional except those on study area, which are 2-dimensional.

Physical measurement units can vary. For example, off-transect distances can be meters ("m"), w.hi can be inches ("in"), and w.lo can be kilometers ("km"). Internally, all distances are converted to the units specified by outputUnits (or the units of input distances if outputUnits is NULL), and

all output is reported in units of `outputUnits`. Valid conversions must exist between units or an error is thrown. For example, meters cannot be converted into hectares.

Measurement units can be assigned using `units()<-` after attaching the `units` package or with `x <- units::set_units(x, "<units>")`. See `units::valid_udunits()` for a list of valid symbolic units.

If measurements are truly unit-less, or measurement units are unknown, set `options(Rdist_requireUnits = FALSE)`. This suppresses all unit checks and conversions. Users are on their own to make sure inputs are scaled correctly and that output units are known.

References

Buckland, S.T., D.R. Anderson, K.P. Burnham, J.L. Laake, D.L. Borchers, and L. Thomas. (2001) *Introduction to distance sampling: estimating abundance of biological populations*. Oxford University Press, Oxford, UK.

See Also

`abundEstim`, `autoDistSamp`. Likelihood-specific help files (e.g., `halfnorm.like`).

Examples

```
# Load example sparrow data (line transect survey type)
data(sparrowDf)

dfunc <- dfuncEstim(data = sparrowDf
                  , formula = dist ~ 1)
dfunc
plot(dfunc)
```

`dfuncEstim` *dfuncEstim - Estimate a distance-based detection function*

Description

Fits a detection function using maximum likelihood.

Usage

```
dfuncEstim(data, ...)
```

Arguments

`data`	An `RdistDf` data frame. `RdistDf` data frames contain one line per transect and a list-based column. The list-based column contains a data frame with detection information. The detection information data frame on each row contains (at least) distances and group sizes of all targets detected on the transect. Function `RdistDf` creates `RdistDf` data frames from separate transect and detection data frames. `is.RdistDf` checks whether data frames are `RdistDf`'s.

... Arguments passed on to dE.single, dE.multi

formula A standard formula object. For example, dist ~ 1, dist ~ covar1 + covar2). The left-hand side (before ~) is the name of the vector containing off-transect or radial detection distances. The right-hand side contains the names of covariate vectors to fit in the detection function, and potentially group sizes. Covariates can be either detection level or transect level and can appear in data or exist in the global working environment. Regular R scoping rules apply.

likelihood String specifying the likelihood to fit. Built-in likelihoods at present are "halfnorm", "hazrate", and "negexp".

w.lo Lower or left-truncation limit of the distances in distance data. This is the minimum possible off-transect distance. Default is 0. If w.lo is greater than 0, it must be assigned measurement units using units(w.lo) <- "<units>" or w.lo <- units::set_units(w.lo, "<units>"). See examples in the help for set_units.

w.hi Upper or right-truncation limit of the distances in dist. This is the maximum off-transect distance that could be observed. If unspecified (i.e., NULL), right-truncation is set to the maximum of the observed distances. If w.hi is specified, it must have associated measurement units. Assign measurement units using units(w.hi) <- "<units>" or w.hi <- units::set_units(w.hi, "<units>"). See examples in the help for set_units.

expansions A scalar specifying the number of terms in series to compute. Depending on the series, this could be 0 through 5. The default of 0 equates to no expansion terms of any type. No expansion terms are allowed (i.e., expansions is forced to 0) if covariates are present in the detection function (i.e., right-hand side of formula includes something other than 1).

series If expansions > 0, this string specifies the type of expansion to use. Valid values at present are 'simple', 'hermite', and 'cosine'.

x.scl The x coordinate (a distance) at which the detection function will be scaled. g.x.scl can be a distance or the string "max". When x.scl is specified (i.e., not 0 or "max"), it must have measurement units assigned using either library(units);units(x.scl) <- '<units>' or x.scl <- units::set_units(x.scl, <units>). See units::valid_udunits() for valid symbolic units.

g.x.scl Height of the distance function at coordinate x. The distance function will be scaled so that g(x.scl) = g.x.scl. If g.x.scl is not a data frame, it must be a numeric value (vector of length 1) between 0 and 1.

warn A logical scalar specifying whether to issue an R warning if the estimation did not converge or if one or more parameter estimates are at their boundaries. For estimation, warn should generally be left at its default value of TRUE. When computing bootstrap confidence intervals, setting warn = FALSE turns off annoying warnings when an iteration does not converge. Regardless of warn, after completion all messages about convergence and boundary conditions are printed by print.dfunc, print.abund, and plot.dfunc.

outputUnits A string specifying the symbolic measurement units for results. Valid units are listed in units::valid_udunits(). The strings for common distance symbolic units are: "m" - meters, "ft" - feet, "cm" - centime-

ters, "mm" - millimeters, "mi" - miles, "nmile" - nautical miles ("nm" is nano meters), "in" - inches, "yd" - yards, "km" - kilometers, "fathom" - fathoms, "chains" - chains, and "furlong" - furlongs. If `outputUnits` is unspecified (NULL), output units will be the same as those on distances in `data`.

Details

Optimization and estimation controls can be modified using `options()`. See `RdistanceControls`.

Value

An object of class 'dfunc'. Objects of class 'dfunc' are lists containing the following components:

`par`	The vector of estimated parameter values. Length of this vector for built-in likelihoods is one (for the function's parameter) plus the number of expansion terms plus one if the likelihood is 'hazrate' (which has two parameters).
`varcovar`	The variance-covariance matrix for coefficients of the distance function, estimated by the inverse of the fit's Hessian evaluated at the estimates. Rdistance estimates the Hessian as the second derivative of the log likelihood surface at the final estimates, where second derivatives are estimated by numeric differentiation (see `secondDeriv`. There is no guarantee this matrix is positive-definite and should be viewed with caution. Error estimates derived from bootstrapping are generally more reliable. I.e., re-compute coefficient confidence intervals using the bootstrap values in component `$B` of an abundance object.
`loglik`	The maximized value of the log likelihood.
`convergence`	The convergence code. This code is returned by `optim` or `nlminb`. Values other than 0 indicate suspect convergence.
`likelihood`	The name of the likelihood. This is the value of the argument `likelihood`.
`w.lo`	Left-truncation value used during the fit.
`w.hi`	Right-truncation value used during the fit.
`mf`	A modelframe of detections within the strip or circle used in the fit. Column 'dist' contains the observed distances. Column 'offset(...)' contains group sizes associated with the values of 'dist'. Group sizes are only used in `abundEstim`. This model frame contains only non-missing distances between `w.lo` and `w.hi`.
`model.frame`	A `model.frame` object containing observed distances (the 'response'), covariates specified in the formula, and group sizes if they were specified. If specified, the name of the group size column is "offset(-variable-)", not "groupsize(-variable-)", because internally it is easier to treat group sizes as an offset in the model. This component is a proper `model.frame` and contains both 'terms' and 'contrasts' attributes.
`siteID.cols`	A vector containing the transect ID column names in `detectionData` and `siteData`. Transect IDs can be a composite of two or more columns and hence this component can have length greater than 1.
`expansions`	The number of expansion terms used during estimation.
`series`	The type of expansion used during estimation.

call	The original call of this function.
call.x.scl	The *input* or user requested distance at which the distance function is scaled.
call.g.x.scl	The input value specifying the height of the distance function at a distance of call.x.scl.
call.observer	The value of input parameter observer. The input observer parameter is only applicable when g.x.scl is a data frame.
fit	The fitted object returned by optim. See documentation for optim.
factor.names	The names of any factors in formula.
pointSurvey	The input value of pointSurvey. This is TRUE if distances are radial from a point. FALSE if distances are perpendicular off-transect.
formula	The formula specified for the detection function.
control	A list containing values of the 'control' parameters set by RdistanceControls.
outputUnits	The measurement units used for output. All distance measurements are converted to these units internally.
x.scl	The *actual* distance at which the distance function is scaled to some value. i.e., this is the actual x at which g(x) = g.x.scl. Note that call.x.scl = x.scl unless call.x.scl == "max", in which case x.scl is the distance at which $g()$ is maximized.
g.x.scl	The *actual* height of the distance function at a distance of x.scl. Note that g.x.scl = call.g.x.scl unless call.g.x.scl is a multiple observer data frame, in which case g.x.scl is the actual height of the distance function at x.scl computed from the multiple observer data frame.

Group Sizes

To specify non-unity group sizes, use groupsize() on the RHS of formula. When group sizes are not all 1, they must appear in a column of the 'detections' list-column of data. For example, d ~ habitat + groupsize(number) specifies distances in column d, one covariate named habitat, and that column number contains the number of individuals associated with each detection. If group sizes are not specified, all group sizes are assumed to be 1.

Contrasts

Factor contrasts in Rdistance are specified the same way as in lm or glm. By default, Rdistance uses contrasts in getOption("contrasts"). To change contrasts, use a statement like options(contrasts = c(unordered = "contr.SAS",ordered = "contr.poly")). Or, to set contrasts for a specific factor in the input data frame, use contrasts(df$A) <- "contr.sum" or similar. See contrasts or the contrasts.arg of model.matrix.

Measurement Units

As of Rdistance version 3.0.0, measurement units are require on all physical distances. Requiring units ensures that internal calculations and results (e.g., ESW and abundance) are correct and that output units are clear. Physical distances are required on off-transect distances, radial distances, truncation distances (w.lo, unless it is zero; and w.hi, unless it is NULL), scale locations (x.scl,

unless it is zero), line-transect lengths, and study area size. All units are 1-dimensional except those on study area, which are 2-dimensional.

Physical measurement units can vary. For example, off-transect distances can be meters ("m"), `w.hi` can be inches ("in"), and `w.lo` can be kilometers ("km"). Internally, all distances are converted to the units specified by `outputUnits` (or the units of input distances if `outputUnits` is NULL), and all output is reported in units of `outputUnits`. Valid conversions must exist between units or an error is thrown. For example, meters cannot be converted into hectares.

Measurement units can be assigned using `units()<-` after attaching the `units` package or with `x <- units::set_units(x, "<units>")`. See `units::valid_udunits()` for a list of valid symbolic units.

If measurements are truly unit-less, or measurement units are unknown, set `options(Rdist_requireUnits = FALSE)`. This suppresses all unit checks and conversions. Users are on their own to make sure inputs are scaled correctly and that output units are known.

References

Buckland, S.T., D.R. Anderson, K.P. Burnham, J.L. Laake, D.L. Borchers, and L. Thomas. (2001) *Introduction to distance sampling: estimating abundance of biological populations*. Oxford University Press, Oxford, UK.

See Also

`abundEstim`, `autoDistSamp`. Likelihood-specific help files (e.g., `halfnorm.like`).

Examples

```
# Sparrow line transect example
data(sparrowDetectionData)
data(sparrowSiteData)

sparrowDf <- RdistDf(sparrowSiteData, sparrowDetectionData)

dfunc <- dfuncEstim(sparrowDf,
                    formula = dist ~ 1
                  )
summary(dfunc)


data(sparrowDfuncObserver) # pre-estimated object
## Not run:
# Command to produce 'sparrowDfuncObserver'
sparrowDfuncObserver <- sparrowDf |>
         dfuncEstim(
           formula = dist ~ observer
         )

## End(Not run)
sparrowDfuncObserver
summary(sparrowDfuncObserver)
plot(sparrowDfuncObserver)
```

dfuncEstimErrMessage *dfuncEstimErrMessage - dfuncEstim error messages*

Description

Utility function to produce error messages suitable for `stop`

Usage

```
dfuncEstimErrMessage(txt, attri)
```

Arguments

`txt`	A text string describing the error.
`attri`	An attribute to report.

Value

A string

distances *distances - Observation distances*

Description

Extract the observation distances (i.e., responses for an Rdistance model) from an Rdistance model frame.

Usage

```
distances(ml, na.rm = TRUE, ...)
```

Arguments

`ml`	Either a Rdistance 'model frame' or an Rdistance 'fitted object'. Both are of class "dfunc". Rdistance 'model frames' are lists containing components necessary to estimate a distance function, but no estimates. Rdistance 'model frames' are typically produced by calls to `parseModel`. Rdistance 'fitted objects' are typically produced by calls to `dfuncEstim`. 'Fitted objects' are 'model frames' with additional components such as the parameters estimates, log likelihood value, convergence information, and the variance- covariance matrix of the parameters.
`na.rm`	Whether to include or exclude missing distance values. In `ml`, the model list containing the model frame, missing values of the response (distance) are potentially present for two reasons: (1) they are outside the strip w.lo to w.hi, and (2) they are missing because the crew did not get a distance for that observation.
`...`	Ignored

Value

A vector containing observation distances contained in the Rdistance model frame.

Examples

```
data(sparrowDf)
sparrowModel <- parseModel( sparrowDf, dist ~ observer )
stats::model.response(sparrowModel$mf)
distances(sparrowModel) # same, but future-proof
```

EDR *EDR - Effective Detection Radius (EDR) for point transects*

Description

Computes Effective Detection Radius (EDR) for estimated detection functions on point transects. See ESW is for line transects.

Usage

```
EDR(object, newdata = NULL)
```

Arguments

object	An Rdistance model frame or fitted distance function, normally produced by a call to dfuncEstim.
newdata	A data frame containing new values of the covariates at which to evaluate the distance functions. If newdata is NULL, distance functions are evaluated at values of the observed covariates and results in one prediction per distance or transect (see parameter type). If newdata is not NULL and the model does not contains covariates, this routine returns one prediction for each row in newdata, but columns and values in newdata are ignored.

Details

Effective Detection Radius is the integral under the detection function times distance. I.e.,

$$EDR = \int_{w.lo}^{w.hi} xg(x)dx,$$

where $g(x)$ is the distance function scaled so that $g(x.scl) = g.x.scl$ and $w.lo$ and $w.hi$ are the lower and upper truncation limits.

Value

If newdata is present, the returned value is a vector of effective sampling distances for values of the covariates in newdata with length equal to the number of rows in newdata. If newdata is NULL, the returned value is a vector of effective sampling distances associated with covariate values in object and has the same number of detected groups. The returned vector has measurement units, i.e., object$outputUnits.

Numeric Integration

Rdistance uses Simpson's composite 1/3 rule to numerically integrate under distance functions. The number of points evaluated during numerical integration is controlled by options(Rdistance_intEvalPts) (default 101). Option 'Rdistance_intEvalPts' must be odd because Simpson's rule requires an even number of intervals (hence, odd number of points). Lower values of 'Rdistance_intEvalPts' increase calculation speeds; but, decrease accuracy. 'Rdistance_intEvalPts' must be >= 5. A warning is thrown if 'Rdistance_intEvalPts' < 29. Empirical tests by the author suggest 'Rdistance_intEvalPts' values >= 30 are accurate to several decimal points and that all 'Rdistance_intEvalPts' >= 101 produce identical results in all but pathological cases.

See Also

dfuncEstim, ESW, effectiveDistance

Examples

```
# Load example thrasher data (point transect survey type)
data(thrasherDf)

# Fit half-normal detection function
dfunc <- thrasherDf |> dfuncEstim(formula=dist~bare)

# Compute effective detection radius (EDR)
EDR(dfunc) # vector length 192
effectiveDistance(dfunc) # same
EDR(dfunc, newdata = data.frame(bare=30)) # vector length 1
```

---

effectiveDistance *effectiveDistance - Effective sampling distances*

Description

Computes Effective Strip Width (ESW) for line-transect detection functions, or the analogous Effective Detection Radius (EDR) for point-transect detection functions.

Usage

```
effectiveDistance(object, newdata = NULL)
```

Arguments

object	An Rdistance model frame or fitted distance function, normally produced by a call to dfuncEstim.
newdata	A data frame containing new values for covariates at which either ESW's or EDR's will be computed. If NULL and object contains covariates, the covariates stored in object are used (like predict.lm). If not NULL, covariate values in newdata are used. See **Value** section for more information.

Details

Serves as a wrapper for ESW and EDR.

Value

If newdata is present, the returned value is a vector of effective sampling distances for values of the covariates in newdata with length equal to the number of rows in newdata. If newdata is NULL, the returned value is a vector of effective sampling distances associated with covariate values in object and has the same number of detected groups. The returned vector has measurement units, i.e., object$outputUnits.

See Also

dfuncEstim ESW EDR

---

effort *effort - Effort information*

Description

Extract effort information from an Rdistance data frame. Effort is length of line-transects or number of points on point-transects.

Usage

```
effort(x, ...)
```

Arguments

x	Either an estimated distance function, output by dfuncEstim, or an Rdistance nested data frame, output by RdistDf.
...	Ignored

Value

A vector containing effort. If line-transects, return is length of transects, with units. If point-transects, return is number of points (integers, no units). Vector length is number of transects. If input is not an RdistDf or estimated distance function, return is NULL.

Examples

```
data(sparrowDf)
effort(sparrowDf)
fit <- dfuncEstim(sparrowDf, dist ~ 1)
effort(fit)
```

`errDataUnk` *errDataUnk - Unknown error message*

Description

Constructs a string stating what is "unknown" that is suitable for use in warning and error functions.

Usage

```
errDataUnk(txt, attri)
```

Arguments

`txt`	Text. The "unknown" we are looking for.
`attri`	Attribute description we are looking for.

Value

A descriptive string, suitable for warning or error.

`estimateN` *estimateN - Abundance point estimates*

Description

Estimate abundance from an Rdistance fitted model. This function is called internally by `abundEstim`. Most users will call `abundEstim` to estimate abundance unless they are running simulations or bootstrapping.

Usage

```
estimateN(object, area = NULL, propUnitSurveyed = 1)
```

Arguments

object	An Rdistance model frame or fitted distance function, normally produced by a call to dfuncEstim.
area	A scalar containing the total area of inference. Usually, this is study area size. If area is NULL (the default), area will be set to 1 square unit of the output units and density estimates will be produced. If area is not NULL, it must have measurement units assigned by the units package. The units on area must be convertible to squared output units. Units on area must be two-dimensional. For example, if output units are "foo", units on area must be convertible to "foo^2" by the units package. Units of "km^2", "cm^2", "ha", "m^2", "acre", "mi^2", and several others are acceptable.
propUnitSurveyed	A scalar or vector of real numbers between 0 and 1. The proportion of the default sampling unit that was surveyed. If both sides of line transects were observed, propUnitSurveyed = 1. If only a single side of line transects were observed, set propUnitSurveyed = 0.5. For point transects, this should be set to the proportion of each circle that was observed. Length must either be 1 or the total number of transects in x.

Details

The abundance estimate for line-transect surveys (if no covariates are included in the detection function and both sides of the transect are observed) is

$$N = \frac{n(A)}{2(ESW)(L)}$$

where *n* is total number of sighted individuals (i.e., sum(groupSizes(dfunc))), *L* is the total length of surveyed transect (i.e., sum(effort(dfunc))), and *ESW* is effective strip width computed from the estimated distance function (i.e., ESW(dfunc)). If only one side of transects were observed, the "2" in the denominator is not present (or, replaced with a "1").

The abundance estimate for point transect surveys (if no covariates are included) is

$$N = \frac{n(A)}{\pi(ESR^2)(P)}$$

where *n* is total number of sighted individuals (i.e., sum(groupSizes(dfunc))), *P* is the total number of surveyed points (i.e., sum(effort(dfunc))), and *ESR* is effective search radius computed from the estimated distance function (i.e., ESR(dfunc)).

Setting plot.bs=FALSE and showProgress=FALSE suppresses all intermediate output.

Estimation of site-specific density (e.g., on every transect) is accomplished by predict(x, type = "density"), which returns a tibble containing density and abundance on the area surveyed by every transect.

Value

A list containing the following components:

density	Estimated density in the surveyed area.

abundance	Estimated abundance on the study area. Equals density if area is not specified.
n.groups	The number of detected groups (not individuals, unless all group sizes = 1).
n.seen	The number of individuals (sum of group sizes).
area	Total area of inference. Study area size
surveyedUnits	Number of surveyed sites. This is total transect length for line-transects or number of points for point-transects. This total transect length does not include transects with missing lengths.
propUnitSurveyed	Proportion of the standard survey unit that was observed
avg.group.size	Average group size on non-NA transects
w	Strip width.
pDetection	Probability of detection.

For line-transects that do not involve covariates, object$density is object$n.seen / (2 * propUnitSurveyed * object$w * object$pDetection * object$surveyedUnits)

See Also

dfuncEstim, abundEstim

ESW *ESW - Effective Strip Width (ESW) for line transects*

Description

Returns effective strip width (ESW) for line-transect detection functions. See EDR is for point transects.

Usage

```
ESW(object, newdata = NULL)
```

Arguments

object	An Rdistance model frame or fitted distance function, normally produced by a call to dfuncEstim.
newdata	A data frame containing new values for covariates at which either ESW's or EDR's will be computed. If NULL and object contains covariates, the covariates stored in object are used (like predict.lm). If not NULL, covariate values in newdata are used. See **Value** section for more information.

Details

ESW is the area under the scaled distance function between its left-truncation limit (`obj$w.lo`) and its right-truncation limit (`obj$w.hi`). I.e.,

$$ESW = \int_{w.lo}^{w.hi} g(x)dx,$$

where $g(x)$ is the distance function scaled so that $g(x.scl) = g.x.scl$ and $w.lo$ and $w.hi$ are the lower and upper truncation limits.

If detection does not decline with distance, the detection function is flat (horizontal), and area under the detection function is $g(0)(w.hi - w.lo)$. If, in this case, $g(0) = 1$, effective sampling distance is the half-width of the surveys, $(w.hi - w.lo)$

Value

If `newdata` is present, the returned value is a vector of effective sampling distances for values of the covariates in `newdata` with length equal to the number of rows in `newdata`. If `newdata` is NULL, the returned value is a vector of effective sampling distances associated with covariate values in `object` and has the same number of detected groups. The returned vector has measurement units, i.e., `object$outputUnits`.

Numeric Integration

Rdistance uses Simpson's composite 1/3 rule to numerically integrate under distance functions. The number of points evaluated during numerical integration is controlled by `options(Rdistance_intEvalPts)` (default 101). Option 'Rdistance_intEvalPts' must be odd because Simpson's rule requires an even number of intervals (hence, odd number of points). Lower values of 'Rdistance_intEvalPts' increase calculation speeds; but, decrease accuracy. 'Rdistance_intEvalPts' must be >= 5. A warning is thrown if 'Rdistance_intEvalPts' < 29. Empirical tests by the author suggest 'Rdistance_intEvalPts' values >= 30 are accurate to several decimal points and that all 'Rdistance_intEvalPts' >= 101 produce identical results in all but pathological cases.

See Also

`dfuncEstim`, `EDR`, `effectiveDistance`

Examples

```
data(sparrowDf)
dfunc <- sparrowDf |> dfuncEstim(formula=dist~bare)

ESW(dfunc) # vector length 356 = number of groups
ESW(dfunc, newdata = data.frame(bare = c(30,40))) # vector length 2
```

`expansionTerms` *expansionTerms - Distance function expansion terms*

Description

Compute "expansion" terms that modify the shape of a base distance function.

Usage

```
expansionTerms(a, d, series, nexp, w)
```

Arguments

`a`	A vector or matrix of (estimated) coefficients. `a` has length p + `nexp` (if a vector) or dimension (k, p + `nexp`), where p is the number of canonical parameters in the likelihood and k is the number of coefficient vectors to evaluate. The first p elements of `a`, or the first p columns if `a` is a matrix, are ignored. I.e., Expansion term coefficients are the last `nexp` elements or columns of `a`.
`d`	A vector or 1-column matrix of distances at which to evaluate the expansion terms. `d` should be distances above w.lo, i.e., distances - w.lo. Parameters `d` and `w` must have compatible measurement units.
`series`	If `expansions` > 0, this string specifies the type of expansion to use. Valid values at present are 'simple', 'hermite', and 'cosine'.
`nexp`	Number of expansion terms. Integer from 0 to 5.
`w`	Strip width, i.e., `w.hi - w.low` = range of `d`. Parameters `d` and `w` must have compatible measurement units.

Details

Expansion terms modify the "key" function of the likelihood manipulatively. The modified distance function is, `key * expTerms` where `key` is a vector of values in the base likelihood function (e.g., `halfnorm.like()$L.unscaled` or `hazrate.like()$L.unscaled`) and `expTerms` is the matrix returned by this routine.

Let the number of expansions (`nexp`) be m (m > 0), assume the raw cyclic expansion terms of `series` are $h_j(x)$ for the j^{th} expansion of distance x, and that $a_1, a_2, \ldots, a_m$ are (estimated) coefficients for the expansion terms, then the likelihood contribution for the i^{th} distance x_i is,

$$f(x_i|\beta, a_1, a_2, \ldots, a_m) = f(x_i|\beta)(1 + \sum_{k=1}^{m} a_k h_k(x_i/w)).$$

Value

If `nexp` equals 0, 1 is returned. If `nexp` is greater than 0, a matrix of size nXk containing expansion terms, where n = `length(d)` and k = `nrow(a)`. The expansion series associated with row j of `a` are

in column j of the return. i.e., element (i,j) of the return is

$$1 + \sum_{k=1}^{m} a_{jk} h_k(x_i/w).$$

(see Details).

Examples

```
a1 <- c(log(40), 0.5, -.5)
a2 <- c(log(40), 0.25, -.5)
dists <- units::set_units(seq(0, 100, length = 100), "m")
w = units::set_units(100, "m")

expTerms1 <- expansionTerms(a1, dists, "cosine", 2, w)
expTerms2 <- expansionTerms(a2, dists, "cosine", 2, w)
plot(dists, expTerms2, ylim = c(0,2.5))
points(dists, expTerms1, pch = 16)

# Same as above
a <- rbind(a1, a2)
expTerms <- expansionTerms(a, dists, "cosine", 2, w)
matlines(dists, expTerms, lwd=2, col=c("red", "blue"), lty=1)

# Showing key and expansions
key <- halfnorm.like(log(40), dists, 1)$L.unscaled
plot(dists, key, type = "l", col = "blue", ylim=c(0,1.5))
lines(dists, key * expTerms1, col = "red")
lines(dists, key * expTerms2, col = "purple")
```

`groupSizes` *groupSizes - Group Sizes*

Description

Extract the group size information from an Rdistance model frame.

Usage

```
groupSizes(ml, ...)
```

Arguments

`ml`	Either a Rdistance 'model frame' or an Rdistance 'fitted object'. Both are of class "dfunc". Rdistance 'model frames' are lists containing components necessary to estimate a distance function, but no estimates. Rdistance 'model frames' are typically produced by calls to `parseModel`. Rdistance 'fitted objects' are typically produced by calls to `dfuncEstim`. 'Fitted objects' are 'model frames'

with additional components such as the parameters estimates, log likelihood value, convergence information, and the variance- covariance matrix of the parameters.

... Ignored

Value

A vector containing group sizes contained in the Rdistance model frame or fitted object.

Examples

```
data("sparrowDf")
sparrowModel <- parseModel( sparrowDf, dist ~ observer )
stats::model.offset(sparrowModel$mf)
groupSizes(sparrowModel)  # same, but future-proof

sparrowModel <- parseModel( sparrowDf
                 , dist ~ observer + groupsize(groupsize) )
groupSizes(sparrowModel)
```

---

gxEstim *gxEstim - Estimate g(0) or g(x)*

Description

Estimate distance function scaling factor , g(0) or g(x), for a specified distance function.

Usage

```
gxEstim(fit)
```

Arguments

`fit`	An estimated `dfunc` object. See `dfuncEstim`.

Details

This routine scales sightability such that g(`x.scl`) = `g.x.scl`, where g() is the sightability function. Specification of `x.scl` and `g.x.scl` covers several estimation cases:

1. **g(0) = 1** : (the default) Inputs are `x.scl` = 0, `g.x.scl` = 1. If `w.lo` > 0, `x.scl` will be set to `w.lo` so technically this case is g(`w.low`) = 1.
2. **User supplied probability at specified distance**: Inputs are `x.scl` = a number greater than or equal to `w.lo`, `g.x.scl` = a number between 0 and 1. This case covers situations where sightability on the transect (distance 0) is not perfect. This case assumes researchers have an independent estimate of sightability at distance `x.scl` off the transect. For example, researchers could be using multiple observers to estimate that sightability at distance `x.scl` is `g.x.scl`.

3. **Maximum sightability specified**: Inputs are `x.scl`="max", `g.x.scl` = a number between 0 and 1. In this case, g() is scaled such that its maximum value is `g.x.scl`. This routine computes the distance at which g() is maximum, sets g()'s height there to `g.x.scl`, and returns `x.max` where x.max is the distance at which g is maximized. This case covers the common aerial survey situation where maximum sightability is slightly off the transect, but the distance at which the maximum occurs is unknown.
4. **Double observer system**: Inputs are `x.scl`="max", `g.x.scl` = <a data frame>. In this case, g(x) = h, where x is the distance that maximizes g and h is the height of g() at x computed from the double observer data frame (see below for structure of the double observer data frame).
5. **Distance of independence specified, height computed from double observer system**: Inputs are `x.scl` = a number greater than or equal to `w.lo` `g.x.scl` = a data frame. In this case, g(`x.scl`) = h, where h is computed from the double observer data frame (see below for structure of the double observer data frame).

When `x.scl`, `g.x.scl`, or `observer` are NULL, the routine will look for and use `$call.x.scl`, or `$call.g.x.scl`, or `$call.observer` components of the `fit` object for whichever of these three parameters is missing. Later, different values can be specified in a direct call to `F.gx.estim` without having to re-estimate the distance function. Because of this feature, the default values in `dfuncEstim` are `x.scl` = 0 and `g.x.scl` = 1 and `observer` = "both".

Value

A list comprised of the following components:

`x.scl`	The value of x (distance) at which g() is evaluated.
`comp2`	The estimated value of g() when evaluated at `x.scl`.

Structure of the double observer data frame

When `g.x.scl` is a data frame, it is assumed to contain the components `$obsby.1` and `$obsby.2` (no flexibility on names). Each row in the data frame contains data from one sighted target. The `$obsby.1` and `$obsby.2` components are TRUE/FALSE (logical) vectors indicating whether observer 1 (`obsby.1`) or observer 2 (`obsby.2`) spotted the target.

See Also

`dfuncEstim`

Examples

```
data(sparrowDf)
fit <- dfuncEstim(sparrowDf, dist ~ groupsize(groupsize))
gxEstim(fit)

fit <- dfuncEstim(sparrowDf, dist ~ groupsize(groupsize)
                , x.scl = units::set_units(50,"m")
                , g.x.scl = 0.75)
gxEstim(fit)
plot(fit)
abline(h=0.75)
```

```
abline(v=units::set_units(50,"m"))
```

`halfnorm.like` *halfnorm.like - Half-normal distance function*

Description

Evaluate the half-normal distance function, for sighting distances, potentially including covariates and expansion terms

Usage

```
halfnorm.like(a, dist, covars)
```

Arguments

`a`	A vector or matrix of covariate and expansion term coefficients. Dimension is k X p, where k (i.e., `nrow(a)`) is the number of coefficient vectors to evaluate (cases) and p (i.e., `ncol(a)`) is the number of covariate and expansion coefficients in the likelihood. If `a` is a dimensionless vector, it is interpreted to be a single row with k = 1. Covariate coefficients in `a` are the first q values (q <= p), and must be on a log scale.
`dist`	A numeric vector of length n or a single-column matrix (dimension nX1) containing detection distances at which to evaluate the likelihood.
`covars`	A numeric vector of length q or matrix of dimension nXq containing covariate values associated with distances in argument `d`

Details

The half-normal distance function is

$$f(d|s) = \exp(-d^2/(2 * s^2))$$

where $s = exp(x'a)$, x is a vector of covariate values associated with distance d (i.e., a row of `covars`), and a is a vector of the first q (=`ncol(covars)`) values in argument `a`.

Some authors parameterize the halfnorm without the "2" in the denominator of the exponent. `Rdistance` includes "2" in this denominator to make quantiles of the half normal agree with the standard normal. This means that half-normal coefficients in Rdistance (i.e., $s = exp(x'a)$) can be interpreted as normal standard errors. For example, approximately 95% of distances should occur between 0 and 2s.

Value

A list containing the following two components:

- **L.unscaled**: A matrix of size nXkXb containing likelihood values evaluated at distances in `dist`. Each row is associated with a single distance, and each column is associated with a single case (row of `a`). This matrix is "unscaled" because the underlying likelihood does not integrate to one. Values in `L.unscaled` are always greater than or equal to zero.
- **params**: A nXkXb array of the likelihood's (canonical) parameters, First page contains parameter values related to covariates (i.e., $s = exp(x'a)$), while subsequent pages contain other parameters. b = 1 for halfnorm, negexp; b = 2 for hazrate and others. Rows correspond to distances in `dist`. Columns correspond to rows from argument `a`.

See Also

`dfuncEstim`, `hazrate.like`, `negexp.like`

Examples

```
d <- seq(0, 100, length=100)
covs <- matrix(1,length(d),1)
halfnorm.like(log(20), d, covs)

plot(d, halfnorm.like(log(20), d, covs)$L.unscaled, type="l", col="red")
lines(d, halfnorm.like(log(40), d, covs)$L.unscaled, col="blue")

# Matrix inputs:
d <- matrix(c(0,10,20), ncol = 1) # 3X1
covs <- matrix(c(rep(1,nrow(d)), rep(.5,nrow(d))), nrow = nrow(d)) # 3X2
coefs <- matrix(log(c(15,5,10,10)), nrow=2) # 2X2
L <- halfnorm.like( coefs, d, covs )
L$L.unscaled # 3X2
L$params     # 3X2; exp(log(15)+0.5log(10)) and exp(log(5)+0.5log(10))
```

---

halfnorm.start.limits *halfnorm.start.limits - Start and limit values for halfnorm distance function*

Description

Compute starting values and limits for the half normal distance function.

Usage

```
halfnorm.start.limits(ml)
```

Arguments

ml	Either a Rdistance 'model frame' or an Rdistance 'fitted object'. Both are of class "dfunc". Rdistance 'model frames' are lists containing components necessary to estimate a distance function, but no estimates. Rdistance 'model frames' are typically produced by calls to parseModel. Rdistance 'fitted objects' are typically produced by calls to dfuncEstim. 'Fitted objects' are 'model frames' with additional components such as the parameters estimates, log likelihood value, convergence information, and the variance- covariance matrix of the parameters.

Value

A list containing the following components

start	Vector of starting values for parameters of the likelihood and expansion terms.
lowlimit	Vector of lower limits for the likelihood parameters and expansion terms.
uplimit	Vector of upper limits for the likelihood parameters and expansion terms.
names	Vector of names for the likelihood parameters and expansion terms.

The length of each vector in the return is: (Num expansions) + 1 + 1*(like %in% c("hazrate")) + (Num Covars).

---

hazrate.like *hazrate.like - Hazard rate likelihood*

Description

Computes the hazard rate distance function.

Usage

```
hazrate.like(a, dist, covars)
```

Arguments

a	A vector or matrix of covariate and expansion term coefficients. Dimension is k X p, where k (i.e., nrow(a)) is the number of coefficient vectors to evaluate (cases) and p (i.e., ncol(a)) is the number of covariate and expansion coefficients in the likelihood. If a is a dimensionless vector, it is interpreted to be a single row with k = 1. Covariate coefficients in a are the first q values (q <= p), and must be on a log scale.
dist	A numeric vector of length n or a single-column matrix (dimension nX1) containing detection distances at which to evaluate the likelihood.
covars	A numeric vector of length q or matrix of dimension nXq containing covariate values associated with distances in argument d

Details

The hazard rate likelihood is

$$f(x|\sigma, k) = 1 - \exp(-(x/\sigma)^{-k})$$

where σ determines location (i.e., distance at which the function equals 1 - exp(-1) = 0.632), and k determines slope of the function at σ (i.e., larger k equals steeper slope at σ). For distance analysis, the valid range for both σ and k is ≥ 0.

Value

A list containing the following two components:

- **L.unscaled**: A matrix of size nXkXb containing likelihood values evaluated at distances in `dist`. Each row is associated with a single distance, and each column is associated with a single case (row of `a`). This matrix is "unscaled" because the underlying likelihood does not integrate to one. Values in `L.unscaled` are always greater than or equal to zero.
- **params**: A nXkXb array of the likelihood's (canonical) parameters, First page contains parameter values related to covariates (i.e., $s = exp(x'a)$), while subsequent pages contain other parameters. b = 1 for halfnorm, negexp; b = 2 for hazrate and others. Rows correspond to distances in `dist`. Columns correspond to rows from argument `a`.

See Also

`dfuncEstim`, `hazrate.like`, `negexp.like`

Examples

```
d <- seq(0, 100, length=100)
covs <- matrix(1,length(d),1)
hazrate.like(c(log(20), 5), d, covs)

# Changing location parameter
plot(d, hazrate.like(c(log(20), 5), d, covs)$L.unscaled, type="l", col="red")
lines(d, hazrate.like(c(log(40), 5), d, covs)$L.unscaled, col="blue")
abline(h = 1 - exp(-1), lty = 2)
abline(v = c(20,40), lty = 2)

# Changing slope parameter
plot(d, hazrate.like(c(log(50), 20), d, covs)$L.unscaled, type="l", col="red")
lines(d, hazrate.like(c(log(50), 2), d, covs)$L.unscaled, col="blue")
abline(h = 1 - exp(-1), lty = 2)
abline(v = 50, lty = 2)
```

`hazrate.start.limits` *hazrate.start.limits - Start and limit values for hazrate distance function*

Description

Compute starting values and limits for the hazard rate distance function.

Usage

```
hazrate.start.limits(ml)
```

Arguments

`ml`	Either a Rdistance 'model frame' or an Rdistance 'fitted object'. Both are of class "dfunc". Rdistance 'model frames' are lists containing components necessary to estimate a distance function, but no estimates. Rdistance 'model frames' are typically produced by calls to `parseModel`. Rdistance 'fitted objects' are typically produced by calls to `dfuncEstim`. 'Fitted objects' are 'model frames' with additional components such as the parameters estimates, log likelihood value, convergence information, and the variance- covariance matrix of the parameters.

Value

A list containing the following components

`start`	Vector of starting values for parameters of the likelihood and expansion terms.
`lowlimit`	Vector of lower limits for the likelihood parameters and expansion terms.
`uplimit`	Vector of upper limits for the likelihood parameters and expansion terms.
`names`	Vector of names for the likelihood parameters and expansion terms.

The length of each vector in the return is: `(Num expansions) + 1 + 1*(like %in% c("hazrate")) + (Num Covars)`.

`hermite.expansion` *Calculation of Hermite expansion for detection function likelihoods*

Description

Computes the Hermite expansion terms used in the likelihood of a distance analysis. More generally, will compute a Hermite expansion of any numeric vector.

Usage

```
hermite.expansion(x, expansions)
```

Arguments

x	In a distance analysis, x is a numeric vector containing the proportion of a strip transect's half-width at which a group of individuals was sighted. If w is the strip transect half-width or maximum sighting distance, and d is the perpendicular off-transect distance to a sighted group ($d \leq w$), x is usually d/w. More generally, x is a vector of numeric values.
expansions	A scalar specifying the number of expansion terms to compute. Must be one of the integers 1, 2, 3, or 4.

Details

There are, in general, several expansions that can be called Hermite. The Hermite expansion used here is:

- **First term**:
$$h_1(x) = x^4 - 6x^2 + 3,$$
- **Second term**:
$$h_2(x) = x^6 - 15x^4 + 45x^2 - 15,$$
- **Third term**:
$$h_3(x) = x^8 - 28x^6 + 210x^4 - 420x^2 + 105,$$
- **Fourth term**:
$$h_4(x) = x^10 - 45x^8 + 630x^6 - 3150x^4 + 4725x^2 - 945,$$

The maximum number of expansion terms computed is 4.

Value

A matrix of size length(x) X expansions. The columns of this matrix are the Hermite polynomial expansions of x. Column 1 is the first expansion term of x, column 2 is the second expansion term of x, and so on up to expansions.

See Also

dfuncEstim, cosine.expansion, simple.expansion, and the discussion of user defined likelihoods in dfuncEstim.

Examples

```
x <- seq(0, 1, length = 200)
herm.expn <- hermite.expansion(x, 4)
plot(range(x), range(herm.expn), type="n")
matlines(x, herm.expn, col=rainbow(4), lty = 1)
```

`intercept.only` *intercept.only - Detect intercept-only distance function*

Description

Utility function to detect whether a distance function has covariates beyond the intercept. If the model contains an intercept-only, effective distance is constant across detections and short-cuts can be implemented in code.

Usage

```
intercept.only(object)
```

Arguments

`object`	An Rdistance model frame or fitted distance function, normally produced by a call to `dfuncEstim`.

Value

TRUE if `object` contains an intercept-only. FALSE if `object` contains at least one detection-level or transect-level covariate in the detection function.

`is.points` *is.points - Tests for point surveys*

Description

Determines whether a distance function is for a point survey or line survey.

Usage

```
is.points(x)
```

Arguments

`x`	Either an estimated distance function, output by `dfuncEstim`, or an Rdistance nested data frame, output by `RdistDf`.

Value

TRUE if the model frame or fitted distance function contains point surveys. FALSE if the model frame or distance function contains line transect surveys.

`is.RdistDf` *checkRdistDf - Check RdistDf data frames*

Description

Checks the validity of `Rdistance` nested data frames. `Rdistance` data frames are a particular implementation of rowwise `tibbles` that contain detections in a list column, and extra attributes specifying types.

Usage

```
is.RdistDf(df, verbose = FALSE)
```

Arguments

`df`	A data frame to check
`verbose`	If TRUE, an explanation of the check that fails is printed. Otherwise, no information on checks is provided.

Details

The following checks are performed (in this order):

- `attr(df, "detectionColumn")` exists and points to a valid list-based column in the data frame.
- `attr(df, "obsType")` exists and is one of the valid values.
- `attr(df, "transType")` exists and is one of the valid values.
- The data frame is either a 'rowwise_df' or 'grouped_df' `tibble`.
- The data frame has only one row per group. One row per group is implied by 'rowwise_df', but not a 'grouped_df', and both are allowed in `Rdistance`. One row per group ensures rows are uniquely identified and hence represents one transect.
- No column names in the list-column are duplicated in the non-list columns of the data frame. This check ensures that `tidyr::unnest` executes.

Other data checks, e.g., for measurement units, are performed later in `dfuncEstim`, after the model is specified.

Value

TRUE or FALSE invisibly. TRUE means all checks passed. FALSE implies at least one check failed. Use `verbose = TRUE` to see which.

Examples

```
data(sparrowDf)
is.RdistDf(sparrowDf)

# Data frame okay, but no attributes
data(sparrowDetectionData)
data(sparrowSiteData)
sparrowDf <- sparrowDetectionData |>
  dplyr::nest_by( siteID
                , .key = "distances") |>
  dplyr::right_join(sparrowSiteData, by = "siteID")
is.RdistDf(sparrowDf, verbose = TRUE)
```

`is.smoothed` *is.smoothed - Tests for smoothed distance functions*

Description

Determines whether a distance function is a non-parametric smooth or classic parameterized function.

Usage

```
is.smoothed(object)
```

Arguments

`object`	An Rdistance model frame or fitted distance function, normally produced by a call to `dfuncEstim`.

Value

TRUE if the model frame or fitted distance function arises from a non-parametric density smoother. FALSE if the model frame or distance function is a parameterized function.

`is.Unitless` *is.Unitless - Test whether object is unitless*

Description

Tests whether a 'units' object is actually unitless. Unitless objects, such as ratios, should be assigned units of '[1]'. Often they are, but sometimes unitless ratios are assigned units like '[m/m]'. The `units` package should always convert '[m/m]' to '[1]', but it does not always. Sometimes units like '[m/m]' mess things up, so it is better to remove them before calculations.

Usage

```
is.Unitless(obj)
```

Arguments

obj	A numeric scalar or vector, with or without units.

Value

TRUE if obj has units and they are either '[1]' or the denominator units equal the numerator units. Otherwise, return FALSE. If obj does not have units, this routine returns TRUE.

Examples

```
a <- units::set_units(2, "m")
b <- a / a
is.Unitless(a)
is.Unitless(b)
is.Unitless(3)
```

---

likeParamNames *Likelihood parameter names*

Description

Returns names of the likelihood parameters. This is a helper function and is not necessary for estimation. It is a nice to label some outputs in Rdistance with parameter names like "sigma" or "knee", depending on the likelihood, and this routine provides a way to do that.

Usage

```
likeParamNames(like.form)
```

Arguments

like.form	A text string naming the form of the likelihood.

Details

For user defined functions, ensure that the user defined start-limits function named <likelihood>.start.limits can be evaluated on a distance of 1, can accept 0 expansions, a low limit of 0 a high limit of 1, and that it returns the parameter names as the $names component of the result. That is, the code that returns user-defined parameter names is, `fn <- match.fun( paste0(like.form, ".start.limits")); ans <- fn(1, 0, 0, 1); ans$names`

Value

A vector of parameter names for that likelihood

lines.dfunc *lines.dfunc - Line plotting method for distance functions*

Description

Line plot method for objects of class 'dfunc' that adds distance functions to an existing plot.

Usage

```
## S3 method for class 'dfunc'
lines(x, newdata = NULL, prob = NULL, ...)
```

Arguments

x	An estimated detection function object, normally produced by calling dfuncEstim.
newdata	A data frame containing new values of the covariates at which to evaluate the distance functions. If newdata is NULL, distance functions are evaluated at values of the observed covariates and results in one prediction per distance or transect (see parameter type). If newdata is not NULL and the model does not contains covariates, this routine returns one prediction for each row in newdata, but columns and values in newdata are ignored.
prob	Logical scalar for whether to scale the distance function to be a density function (integrates to one). Default behavior is designed to be compatible with the plot method for distance functions (plot.dfunc). By default, line transect distance functions are not scaled to a density and integrate to the effective strip width. In contrast, point transects distance functions are scaled to be densities by default.
...	Parameters passed to lines.default that control attributes like color, line width, line type, etc.

Value

A data frame containing the x and y coordinates of the plotted line(s) is returned invisibly. X coordinates in the return are names x. Y coordinates in the return are named y1, y2, ..., yn, i.e., one column per returned distance function.

See Also

dfuncEstim, plot.dfunc, print.abund

Examples

```
set.seed(87654)
x <- rnorm(1000, mean=0, sd=20)
x <- x[x >= 0]
x <- units::set_units(x, "mi")
Df <- data.frame(transectID = "A"
                 , distance = x
```

```
                ) |>
  dplyr::nest_by( transectID
                , .key = "detections") |>
  dplyr::mutate(length = units::set_units(100,"km"))
attr(Df, "detectionColumn") <- "detections"
attr(Df, "obsType") <- "single"
attr(Df, "transType") <- "line"
attr(Df,'effortColumn') <- "length"
is.RdistDf(Df)  # TRUE

dfunc <- Df |> dfuncEstim(distance ~ 1, likelihood="halfnorm")
plot(dfunc, nbins = 40, col="lightgrey", border=NA, vertLines=FALSE)
lines(dfunc, col="grey30", lwd=15)
lines(dfunc, col="grey90", lwd=5, lty = 2)

# Multiple lines
data(sparrowDfuncObserver)
obsLevs <- levels(sparrowDfuncObserver$data$observer)
plot(sparrowDfuncObserver
   , vertLines = FALSE
   , lty = 0
   , plotBars = FALSE
   , main="Detection by observer"
   , legend = FALSE)
y <- lines(sparrowDfuncObserver
   , newdata = data.frame(observer = obsLevs)
   , col = palette.colors(length(obsLevs))
   , lty = 1
   , lwd = 4)
head(y) # values returned, with distances as column
```

`maximize.g` *maximize.g - Find coordinate of function maximum*

Description

Find the x coordinate that maximizes g(x).

Usage

```
maximize.g(fit, covars = NULL)
```

Arguments

`fit`	An estimated 'dfunc' object produced by `dfuncEstim`.
`covars`	Covariate values to calculate g(x).

Value

The value of x that maximizes g(x) in `fit`.

See Also

`dfuncEstim`

Examples

```
## Not run:
# Fake data
set.seed(22223333)
x <- rgamma(100, 10, 1)

fit <- dfuncEstim( x, likelihood="Gamma", x.scl="max" )

maximize.g( fit )  # should be near 10.
fit$x.scl            # same thing

## End(Not run)
```

`mlEstimates` *mlEstimates - Distance function maximum likelihood estimates*

Description

Estimate parameters of a distance function using maximum likelihood.

Usage

```
mlEstimates(ml, strt.lims)
```

Arguments

`ml`	Either a Rdistance 'model frame' or an Rdistance 'fitted object'. Both are of class "dfunc". Rdistance 'model frames' are lists containing components necessary to estimate a distance function, but no estimates. Rdistance 'model frames' are typically produced by calls to `parseModel`. Rdistance 'fitted objects' are typically produced by calls to `dfuncEstim`. 'Fitted objects' are 'model frames' with additional components such as the parameters estimates, log likelihood value, convergence information, and the variance- covariance matrix of the parameters.
`strt.lims`	A list containing start, low, and high limits for parameters of the requested likelihood. This list is typically produced by a call to `startLimits`.

Value

An Rdistance fitted model object. This object contains the raw object returned by the optimization routine (e.g., `nlming`), and additional components specific to Rdistance.

`model.matrix.dfunc` *model.matrix - Rdistance model matrix*

Description

Extract the model matrix ("X" matrix) from an Rdistance model object.

Usage

```
## S3 method for class 'dfunc'
model.matrix(object, ...)
```

Arguments

`object`	An Rdistance model frame or fitted distance function, normally produced by a call to `dfuncEstim`.
`...`	Ignored

Value

A matrix containing covariates for fitting an Rdistance model.

Examples

```
data(sparrowDf)
sparrowModel <- parseModel( sparrowDf, dist ~ observer )
model.matrix(sparrowModel)
```

`nCovars` *nCovars - Number of covariates*

Description

Return number of covariates in a distance model

Usage

```
nCovars(X)
```

Arguments

X	The X matrix of covariates, or a vector.

Details

The reason this routine is needed is that sometimes we pass one row of covariates to a likelihood function. If so, it may come in as a normal vector, not a matrix. If a normal vector, ncol(X) does not work.

Value

An integer scalar

do not export

---

negexp.like *negexp.like - Negative exponential likelihood*

Description

Computes the negative exponential distance function.

Usage

```
negexp.like(a, dist, covars)
```

Arguments

a	A vector or matrix of covariate and expansion term coefficients. Dimension is k X p, where k (i.e., nrow(a)) is the number of coefficient vectors to evaluate (cases) and p (i.e., ncol(a)) is the number of covariate and expansion coefficients in the likelihood. If a is a dimensionless vector, it is interpreted to be a single row with k = 1. Covariate coefficients in a are the first q values (q <= p), and must be on a log scale.
dist	A numeric vector of length n or a single-column matrix (dimension nX1) containing detection distances at which to evaluate the likelihood.
covars	A numeric vector of length q or matrix of dimension nXq containing covariate values associated with distances in argument d

Details

The negative exponential likelihood is

$$f(x|a) = \exp(-ax)$$

where a is the slope parameter.

Value

A list containing the following two components:

- **L.unscaled**: A matrix of size nXkXb containing likelihood values evaluated at distances in `dist`. Each row is associated with a single distance, and each column is associated with a single case (row of `a`). This matrix is "unscaled" because the underlying likelihood does not integrate to one. Values in `L.unscaled` are always greater than or equal to zero.
- **params**: A nXkXb array of the likelihood's (canonical) parameters, First page contains parameter values related to covariates (i.e., $s = exp(x'a)$), while subsequent pages contain other parameters. b = 1 for halfnorm, negexp; b = 2 for hazrate and others. Rows correspond to distances in `dist`. Columns correspond to rows from argument `a`.

See Also

`dfuncEstim`, `hazrate.like`, `negexp.like`

Examples

```
d <- seq(0, 100, length=100)
covs <- matrix(1,length(d),1)
negexp.like(log(0.01), d, covs)

# Changing slope parameter
plot(d, negexp.like(log(0.1), d, covs)$L.unscaled, type="l", col="red")
lines(d, negexp.like(log(0.05), d, covs)$L.unscaled, col="blue")
```

`negexp.start.limits` *negexp.start.limits - Start and limit values for negexp distance function*

Description

Compute starting values and limits for the negative exponential distance function.

Usage

```
negexp.start.limits(ml)
```

Arguments

`ml`	Either a Rdistance 'model frame' or an Rdistance 'fitted object'. Both are of class "dfunc". Rdistance 'model frames' are lists containing components necessary to estimate a distance function, but no estimates. Rdistance 'model frames' are typically produced by calls to `parseModel`. Rdistance 'fitted objects' are typically produced by calls to `dfuncEstim`. 'Fitted objects' are 'model frames' with additional components such as the parameters estimates, log likelihood value, convergence information, and the variance- covariance matrix of the parameters.

Value

A list containing the following components

`start`	Vector of starting values for parameters of the likelihood and expansion terms.
`lowlimit`	Vector of lower limits for the likelihood parameters and expansion terms.
`uplimit`	Vector of upper limits for the likelihood parameters and expansion terms.
`names`	Vector of names for the likelihood parameters and expansion terms.

The length of each vector in the return is: `(Num expansions) + 1 + 1*(like %in% c("hazrate")) + (Num Covars)`.

`nLL` *nLL - Negative log likelihood of distances*

Description

Return the negative log likelihood of observed detection distances given a likelihood and the estimated parameters.

Usage

```
nLL(a, ml)
```

Arguments

`a`	A vector of likelihood parameter values. Length and meaning depend on `ml$series` and `ml$expansions`. If no expansion terms were called for (i.e., `ml$expansions = 0`), the distance likelihood contain one or two canonical parameters (see Details). If one or more expansions are called for, coefficients for the expansion terms follow coefficients for the canonical parameters. i.e., length of this vector is `(num Covars incl. intercept) + expansions + 1*(like %in% c("hazrate"))`.
`ml`	Either a Rdistance 'model frame' or an Rdistance 'fitted object'. Both are of class "dfunc". Rdistance 'model frames' are lists containing components necessary to estimate a distance function, but no estimates. Rdistance 'model frames' are typically produced by calls to `parseModel`. Rdistance 'fitted objects' are typically produced by calls to `dfuncEstim`. 'Fitted objects' are 'model frames' with additional components such as the parameters estimates, log likelihood value, convergence information, and the variance- covariance matrix of the parameters.

Details

Expansion Terms: If `ml$expansions` = k (k > 0), the expansion function specified by `ml$series` is called (see for example `cosine.expansion`). Assuming $h_{ij}(x)$ is the j^{th} expansion term for the i^{th} distance and that $c_1, c_2, \ldots, c_k$ are (estimated) coefficients for the expansion terms, the likelihood contribution for the i^{th} distance is,

$$f(x|a, b, c_1, c_2, \ldots, c_k) = f(x|a, b)(1 + \sum_{j=1}^{k} c_j h_{ij}(x)).$$

Value

A scalar, the negative of the log likelihood evaluated at parameters `a`.

See Also

See `halfnorm.like` and links there; `dfuncEstim`

Examples

```
set.seed(238642)

d <- rnorm(1000, mean = 0, sd = 40)
d <- units::set_units(d[0 <= d], "m")

# Min info in model list to compute likelihood
ml <- list(
    mf = model.frame(d ~ 1)
  , likelihood = "halfnorm"
  , expansions = 0
  , w.lo = units::set_units(0, "m")
  , w.hi = units::set_units(125, "m")
  , outputUnits = units(units::set_units(1,"m"))
  , x.scl = units::set_units(0,"m")
  , g.x.scl = 1
  , data = 1
)
attr(ml$data, "transType") <- "line"
class(ml) <- "dfunc"
nLL(log(40), ml)

# Another way, b/c we have pnorm()
ones <- matrix(1, nrow = length(d), ncol = 1)
l <- halfnorm.like(log(40), d, ones)
scaler <-(pnorm(units::drop_units(ml$w.hi)
  , units::drop_units(ml$w.lo)
  , sd = l$params) - 0.5) * sqrt(2*pi) * l$params
-sum(log(l$L.unscaled/scaler))

# A third way, b/c we have pnorm() and dnorm().
l2 <- dnorm(units::drop_units(d), mean = 0, sd = 40)
scaler2 <- pnorm(125, mean = 0, sd = 40) - 0.5
-sum(log(l2/scaler2))
```

`observationType` *observationType - Type of observations*

Description

Return the type of observations (single or multiple observers) represented in either a fitted distance function or Rdistance data frame.

Usage

```
observationType(x)
```

Arguments

x	Either an estimated distance function, output by dfuncEstim, or an Rdistance nested data frame, output by RdistDf.

Details

This function is a simple helper function. If x is an estimated distance object, it polls the obsType attribute of the object's Rdistance data frame. If x is an Rdistance nested data frame, it polls the obsType attribute.

Value

One of the following values: "single", "1given2", "2given1", or "both". If observation type has not been assigned, return is NULL.

---

oneBsIter *oneBsIter - Computations for one bootstrap iteration*

Description

An internal (un-exported) function to perform density and abundance calculations on one iteration of the bootstrap.

Usage

```
oneBsIter(
  indexDf,
  key,
  data,
  formula,
  likelihood,
  w.lo,
  w.hi,
  expansions,
  series,
  x.scl,
  g.x.scl,
  outputUnits,
  warn,
  area,
  propUnitSurveyed,
  pb,
  plot.bs,
  plotCovValues
)
```

Arguments

indexDf	A data frame containing row indices to use for subsetting the rows of data. The actual indices are in column rowIndex.
key	A data frame containing the current id of the BS iteration. This is included for compatability with dplyr::group_modify, but it is not used internally. The original non-resampled data have key == "Original".
data	An Rdistance nested data frame containing the data to bootstrap resample. Rows of this data frame, equating to transects, are sampled using the indicies in indexDf$rowIndex.
formula	A standard formula object. For example, dist ~ 1, dist ~ covar1 + covar2). The left-hand side (before ~) is the name of the vector containing off-transect or radial detection distances. The right-hand side contains the names of covariate vectors to fit in the detection function, and potentially group sizes. Covariates can be either detection level or transect level and can appear in data or exist in the global working environment. Regular R scoping rules apply.
likelihood	String specifying the likelihood to fit. Built-in likelihoods at present are "halfnorm", "hazrate", and "negexp".
w.lo	Lower or left-truncation limit of the distances in distance data. This is the minimum possible off-transect distance. Default is 0. If w.lo is greater than 0, it must be assigned measurement units using units(w.lo) <- "<units>" or w.lo <- units::set_units(w.lo, "<units>"). See examples in the help for set_units.
w.hi	Upper or right-truncation limit of the distances in dist. This is the maximum off-transect distance that could be observed. If unspecified (i.e., NULL), right-truncation is set to the maximum of the observed distances. If w.hi is specified, it must have associated measurement units. Assign measurement units using units(w.hi) <- "<units>" or w.hi <- units::set_units(w.hi, "<units>"). See examples in the help for set_units.
expansions	A scalar specifying the number of terms in series to compute. Depending on the series, this could be 0 through 5. The default of 0 equates to no expansion terms of any type. No expansion terms are allowed (i.e., expansions is forced to 0) if covariates are present in the detection function (i.e., right-hand side of formula includes something other than 1).
series	If expansions > 0, this string specifies the type of expansion to use. Valid values at present are 'simple', 'hermite', and 'cosine'.
x.scl	The x coordinate (a distance) at which the detection function will be scaled. g.x.scl can be a distance or the string "max". When x.scl is specified (i.e., not 0 or "max"), it must have measurement units assigned using either library(units);units(x.scl) <- '<units>' or x.scl <- units::set_units(x.scl, <units>). See units::valid_udunits() for valid symbolic units.
g.x.scl	Height of the distance function at coordinate x. The distance function will be scaled so that g(x.scl) = g.x.scl. If g.x.scl is not a data frame, it must be a numeric value (vector of length 1) between 0 and 1.
outputUnits	A string specifying the symbolic measurement units for results. Valid units are listed in units::valid_udunits(). The strings for common distance symbolic units are: "m" - meters, "ft" - feet, "cm" - centimeters, "mm" - millimeters, "mi"

- miles, "nmile" - nautical miles ("nm" is nano meters), "in" - inches, "yd" - yards, "km" - kilometers, "fathom" - fathoms, "chains" - chains, and "furlong" - furlongs. If `outputUnits` is unspecified (NULL), output units will be the same as those on distances in `data`.

`warn` A logical scalar specifying whether to issue an R warning if the estimation did not converge or if one or more parameter estimates are at their boundaries. For estimation, `warn` should generally be left at its default value of `TRUE`. When computing bootstrap confidence intervals, setting `warn = FALSE` turns off annoying warnings when an iteration does not converge. Regardless of `warn`, after completion all messages about convergence and boundary conditions are printed by `print.dfunc`, `print.abund`, and `plot.dfunc`.

`area` A scalar containing the total area of inference. Usually, this is study area size. If `area` is NULL (the default), `area` will be set to 1 square unit of the output units and density estimates will be produced. If `area` is not NULL, it must have measurement units assigned by the `units` package. The units on `area` must be convertible to squared output units. Units on `area` must be two-dimensional. For example, if output units are "foo", units on area must be convertible to "foo^2" by the `units` package. Units of "km^2", "cm^2", "ha", "m^2", "acre", "mi^2", and several others are acceptable.

`propUnitSurveyed`
A scalar or vector of real numbers between 0 and 1. The proportion of the default sampling unit that was surveyed. If both sides of line transects were observed, `propUnitSurveyed = 1`. If only a single side of line transects were observed, set `propUnitSurveyed = 0.5`. For point transects, this should be set to the proportion of each circle that was observed. Length must either be 1 or the total number of transects in `x`.

`pb` A progress bar created with `progress::progress_bar$new()`.

`plot.bs` Logical. Whether to plot bootstrap estimate of detection function. A plot must already exist because this uses `lines`.

`plotCovValues` Data frame containing values of covariates to plot, if `plot.bs` is TRUE.

Value

A data frame containing density and abundance and other relevant statistics for one iteration of the bootstrap.

`parseModel` *parseModel - Parse Rdistance model*

Description

Parse an 'Rdistance' formula and produce a list containing all model parameters.

Usage

```
parseModel(
  data,
  formula = NULL,
  likelihood = "halfnorm",
  w.lo = 0,
  w.hi = NULL,
  expansions = 0,
  series = "cosine",
  x.scl = 0,
  g.x.scl = 1,
  outputUnits = NULL
)
```

Arguments

data	An RdistDf data frame. RdistDf data frames contain one line per transect and a list-based column. The list-based column contains a data frame with detection information. The detection information data frame on each row contains (at least) distances and group sizes of all targets detected on the transect. Function RdistDf creates RdistDf data frames from separate transect and detection data frames. is.RdistDf checks whether data frames are RdistDf's.
formula	A standard formula object. For example, dist ~ 1, dist ~ covar1 + covar2). The left-hand side (before ~) is the name of the vector containing off-transect or radial detection distances. The right-hand side contains the names of covariate vectors to fit in the detection function, and potentially group sizes. Covariates can be either detection level or transect level and can appear in data or exist in the global working environment. Regular R scoping rules apply.
likelihood	String specifying the likelihood to fit. Built-in likelihoods at present are "halfnorm", "hazrate", and "negexp".
w.lo	Lower or left-truncation limit of the distances in distance data. This is the minimum possible off-transect distance. Default is 0. If w.lo is greater than 0, it must be assigned measurement units using units(w.lo) <- "<units>" or w.lo <- units::set_units(w.lo, "<units>"). See examples in the help for set_units.
w.hi	Upper or right-truncation limit of the distances in dist. This is the maximum off-transect distance that could be observed. If unspecified (i.e., NULL), right-truncation is set to the maximum of the observed distances. If w.hi is specified, it must have associated measurement units. Assign measurement units using units(w.hi) <- "<units>" or w.hi <- units::set_units(w.hi, "<units>"). See examples in the help for set_units.
expansions	A scalar specifying the number of terms in series to compute. Depending on the series, this could be 0 through 5. The default of 0 equates to no expansion terms of any type. No expansion terms are allowed (i.e., expansions is forced to 0) if covariates are present in the detection function (i.e., right-hand side of formula includes something other than 1).

series	If expansions > 0, this string specifies the type of expansion to use. Valid values at present are 'simple', 'hermite', and 'cosine'.
x.scl	The x coordinate (a distance) at which the detection function will be scaled. g.x.scl can be a distance or the string "max". When x.scl is specified (i.e., not 0 or "max"), it must have measurement units assigned using either library(units);units(x.scl) <- '<units>' or x.scl <- units::set_units(x.scl, <units>). See units::valid_udunits() for valid symbolic units.
g.x.scl	Height of the distance function at coordinate x. The distance function will be scaled so that g(x.scl) = g.x.scl. If g.x.scl is not a data frame, it must be a numeric value (vector of length 1) between 0 and 1.
outputUnits	A string specifying the symbolic measurement units for results. Valid units are listed in units::valid_udunits(). The strings for common distance symbolic units are: "m" - meters, "ft" - feet, "cm" - centimeters, "mm" - millimeters, "mi" - miles, "nmile" - nautical miles ("nm" is nano meters), "in" - inches, "yd" - yards, "km" - kilometers, "fathom" - fathoms, "chains" - chains, and "furlong" - furlongs. If outputUnits is unspecified (NULL), output units will be the same as those on distances in data.

Details

This routine is not intended to be called by the user. It is called from the model estimation routines in Rdistance.

Value

An Rdistance model frame, which is an object of class "dfunc". Rdistance model frames are lists containing distance model components but not estimates. Model frames contain everything necessary to fit an Rdistance mode, such as covariates, minimum and maximum distances, the form of the likelihood, number of expansions, etc. Rdistance model frames contain a subset of fitted Rdistance model components.

See Also

[RdistDf()], which returns an Rdistance *data* frame; [dfuncEstim()], which returns an Rdistance *fitted* model.

Examples

```
data(sparrowSiteData)
data(sparrowDetectionData)

sparrowDf <- Rdistance::RdistDf(sparrowSiteData
   , sparrowDetectionData
   , by = NULL
   , pointSurvey = FALSE
   , observer = "single"
   , .detectionCol = "detections")

ml <- Rdistance::parseModel(sparrowDf
```

```
   , formula = dist ~ 1 + observer + groupsize(groupsize)
   , likelihood = "halfnorm"
   , w.lo = 0
   , w.hi = NULL
   , series = "cosine"
   , x.scl = 0
   , g.x.scl = 1
   , outputUnits = "m"
   )
class(ml)  # 'dfunc', but no estimated coefficients
print(ml)
print.default(ml)
```

---

perpDists *Compute off-transect distances from sighting distances and angles*

Description

Computes off-transect (also called 'perpendicular') distances from measures of sighting distance and sighting angle.

Usage

```
perpDists(sightDist, sightAngle, data)
```

Arguments

sightDist	Character, name of column in data that contains the observed or sighting distances from the observer to the detected objects.
sightAngle	Character, name of column in data that contains the observed or sighting angles from the line transect to the detected objects. Angles must be measured in degrees.
data	data.frame object containing sighting distance and sighting angle.

Details

If observers recorded sighting distance and sighting angle (as is often common in line transect surveys), use this function to convert to off-transect distances, the required input data for dfunc.estim.

Value

A vector of off-transect (or perpendicular) distances. Units are the same as sightDist.

References

Buckland, S.T., Anderson, D.R., Burnham, K.P. and Laake, J.L. 1993. *Distance Sampling: Estimating Abundance of Biological Populations*. Chapman and Hall, London.

See Also

dfuncEstim

Examples

```
# Load the example dataset of sparrow detections from package
data(sparrowDetectionData)
# Compute perpendicular, off-transect distances from the observer's sight distance and angle
sparrowDetectionData$perpDist <- perpDists(sightDist="sightdist", sightAngle="sightangle",
                                           data=sparrowDetectionData)
```

---

plot.dfunc *plot.dfunc - Plot method for distance (detection) functions*

Description

Plot method for objects of class 'dfunc'. Objects of class 'dfunc' are estimated distance functions produced by dfuncEstim.

Usage

```
## S3 method for class 'dfunc'
plot(x, ...)
```

Arguments

x	An estimated detection function object, normally produced by calling dfuncEstim.
...	Arguments passed on to plot.dfunc.para
include.zero	Boolean value specifying whether to include 0 on the x-axis of the plot. A value of TRUE will include 0 on the left hand end of the x-axis regardless of the range of distances. A value of FALSE will plot only the observation strip (w.lo to w.hi).
nbins	Internally, this function uses hist to compute histogram bars for the plot. This argument is the breaks argument to hist. This can be either a vector giving the breakpoints between bars, the suggested number of bars (a single number), a string naming an algorithm to compute the number of bars, or a function to compute the number of bars. See hist for all options.
newdata	Data frame (similar to newdata parameter of lm) containing new values for covariates in the distance function. One distance function is computed and plotted for each row in the data frame. If newdata is NULL, a single distance function is plotted for mean values of all numeric covariates and mode values for all factor covariates.
legend	Logical scalar for whether to include a legend. If TRUE, a legend will be included on the plot detailing the covariate values used to generate the plot.

`plotBars` Logical scalar for whether to plot the histogram of distances behind the distance function. If FALSE, no histogram is plotted, only the distance function line(s).

`xlab` Label for the x-axis

`ylab` Label for the y-axis

`density` If `plotBars=TRUE`, a vector giving the density of shading lines, in lines per inch, for the bars underneath the distance function, repeated as necessary to exceed the number of bars. Values of NULL or a number strictly less than 0 mean solid fill using colors from parameter `col`. If `density = 0`, bars are not filled and only the borders are rendered. If `density > 0`, bars are shaded with colors and angles from parameters `col` and `angle`.

`angle` When `density > 0`, the slope of bar shading lines, given as an angle in degrees (counter-clockwise), repeated as necessary to exceed the number of bars.

`col` A vector of bar fill colors or line colors when bars are drawn and `density != 0`, repeated as necessary to exceed the number of bars. Also used for the bar borders when `border = TRUE`.

`border` The color of bar borders when bars are plotted, repeated as necessary to exceed the number of bars. A value of NA or FALSE means no borders. If bars are shaded with lines (i.e., `density>0`), `border = TRUE` uses the same color for the border as for the shading lines. Otherwise, fill color or shaded line color are specified in `col` while border color is specified in `border`.

`vertLines` Logical scalar specifying whether to plot vertical lines at `w.lo` and `w.hi` from 0 to the distance function.

`col.dfunc` Color of the distance function(s). If only one distance function (one line) is being plotted, the default color is "red". If covariates or `newdata` are present, the default value uses `graphics::rainbow(n)`, where `n` is the number of plotted distance functions. Otherwise, `col.dfunc` is replicated to the required length. Plot all distance functions in the same color by setting `col.dfunc` to a scalar. Plot blue-red pairs of distance functions by setting `col.dfunc = c("blue", "red")`. Etc.

`lty.dfunc` Line type of the distance function(s). If covariates or `newdata` is present, the default uses line types to `1:n`, where `n` is the number of plotted distance functions. Otherwise, `lty.dfunc` is replicated to the required length. Plot solid lines by specifying `lty.dfunc = 1`. Plot solid-dashed line pairs by specifying `lty.dfunc = c(1,2)`. Etc.

`lwd.dfunc` Line width of the distance function(s), replicated to the required length. Default is 2 for all lines.

Details

If `plotBars` is TRUE, a scaled histogram is plotted and the estimated distance function is plotted over the top of it. When bars are plotted, this routine uses `graphics::barplot` for setting up the initial plotting region and most parameters to `graphics::barplot` can be specified (exceptions noted above in description of '...').

The form of the likelihood and any series expansions is printed in the main title (overwrite this with `main="<my title>"`). Convergence of the distance function is checked. If the distance function did

not converge, a warning is printed over the top of the histogram. If one or more parameter estimates are at their limits (likely indicating non-convergence or poor fit), another warning is printed.

Value

The input distance function is returned, with two additional components than can be used to reconstruct the plotted bars. (To obtain values of the plotted distance functions, use `predict` with `type = "distances"`.) The additional components are:

`barHeights`	A vector containing the scaled bar heights drawn on the plot.
`barWidths`	A vector or scalar of the bar widths drawn on the plot, with measurement units.

Re-plot the bars with `barplot( return$barHeights, width = return$barWidths )`.

See Also

`dfuncEstim`, `print.dfunc`, `print.abund`

Examples

```
set.seed(87654)
x <- rnorm(1000, mean=0, sd=20)
x <- x[x >= 0]
x <- units::set_units(x, "ft")
Df <- data.frame(transectID = "A"
                , distance = x
                 ) |>
  dplyr::nest_by( transectID
                , .key = "detections") |>
  dplyr::mutate(length = units::set_units(1,"mi"))
attr(Df, "detectionColumn") <- "detections"
attr(Df, "obsType") <- "single"
attr(Df, "transType") <- "line"
attr(Df, "effortColumn") <- "length"
is.RdistDf(Df) # TRUE

dfunc <- Df |> dfuncEstim(distance ~ 1, likelihood="halfnorm")
plot(dfunc)
plot(dfunc, nbins=25)

# showing effects of plot params
plot(dfunc
  , col=c("red","blue","orange")
  , border="black"
  , xlab="Off-transect distance"
  , ylab="Prob"
  , vertLines = FALSE
  , main="Showing plot params")

plot(dfunc
   , col="purple"
   , density=30
```

```
   , angle=c(-45,0,45)
   , cex.axis=1.5
   , cex.lab=2
   , ylab="Probability")

plot(dfunc
   , col=c("grey","lightgrey")
   , border=NA)

plot(dfunc
   , col="grey"
   , border=0
   , col.dfunc="blue"
   , lty.dfunc=2
   , lwd.dfunc=4
   , vertLines=FALSE)

plot(dfunc
   , plotBars=FALSE
   , cex.axis=1.5
   , col.axis="blue")
rug(distances(dfunc))

# Plot showing f(0)
hist(distances(dfunc)
   , n = 40
   , border = NA
   , prob = TRUE)
x <- seq(dfunc$w.lo, dfunc$w.hi, length=200)
g <- predict(dfunc, type="dfunc", distances = x, newdata = data.frame(a=1))
f <- g[,1] / ESW(dfunc)[1]
# Check integration:
sum(diff(x)*(f[-1] + f[-length(f)]) / 2) # Trapazoid rule; should be 1.0
lines(x, f) # hence, 1/f(0) = ESW

# Covariates: detection by observer
data(sparrowDfuncObserver) # pre-estimated model

obsLevs <- levels(sparrowDfuncObserver$data$observer)
plot(sparrowDfuncObserver
   , newdata = data.frame(observer = obsLevs)
   , vertLines = FALSE
   , col.dfunc = heat.colors(length(obsLevs))
   , col = c("grey","lightgrey")
   , border=NA
   , main="Detection by observer")
```

---

plot.dfunc.para *plot.dfunc.para - Plot parametric distance functions*

Description

Plot method for parametric line and point transect distance functions.

Usage

```
## S3 method for class 'dfunc.para'
plot(
  x,
  include.zero = FALSE,
  nbins = "Sturges",
  newdata = NULL,
  legend = TRUE,
  vertLines = TRUE,
  plotBars = TRUE,
  circles = FALSE,
  density = -1,
  angle = 45,
  xlab = NULL,
  ylab = NULL,
  border = TRUE,
  col = "grey85",
  col.dfunc = NULL,
  lty.dfunc = NULL,
  lwd.dfunc = NULL,
  ...
)
```

Arguments

x	An estimated detection function object, normally produced by calling dfuncEstim.
include.zero	Boolean value specifying whether to include 0 on the x-axis of the plot. A value of TRUE will include 0 on the left hand end of the x-axis regardless of the range of distances. A value of FALSE will plot only the observation strip (w.lo to w.hi).
nbins	Internally, this function uses hist to compute histogram bars for the plot. This argument is the breaks argument to hist. This can be either a vector giving the breakpoints between bars, the suggested number of bars (a single number), a string naming an algorithm to compute the number of bars, or a function to compute the number of bars. See hist for all options.
newdata	Data frame (similar to newdata parameter of lm) containing new values for covariates in the distance function. One distance function is computed and plotted for each row in the data frame. If newdata is NULL, a single distance function is plotted for mean values of all numeric covariates and mode values for all factor covariates.
legend	Logical scalar for whether to include a legend. If TRUE, a legend will be included on the plot detailing the covariate values used to generate the plot.

vertLines — Logical scalar specifying whether to plot vertical lines at w.lo and w.hi from 0 to the distance function.

plotBars — Logical scalar for whether to plot the histogram of distances behind the distance function. If FALSE, no histogram is plotted, only the distance function line(s).

circles — Logical scalar requesting the location of detection distances be plotted. If TRUE, open circles are plotted at predicted distance function heights associated with all detection distances. For computational simplicity, all distances are plotted for EVERY covariate class even though observed distances belong to only one covariate class. If FALSE, circles are not plotted.

density — If plotBars=TRUE, a vector giving the density of shading lines, in lines per inch, for the bars underneath the distance function, repeated as necessary to exceed the number of bars. Values of NULL or a number strictly less than 0 mean solid fill using colors from parameter col. If density = 0, bars are not filled and only the borders are rendered. If density > 0, bars are shaded with colors and angles from parameters col and angle.

angle — When density > 0, the slope of bar shading lines, given as an angle in degrees (counter-clockwise), repeated as necessary to exceed the number of bars.

xlab — Label for the x-axis

ylab — Label for the y-axis

border — The color of bar borders when bars are plotted, repeated as necessary to exceed the number of bars. A value of NA or FALSE means no borders. If bars are shaded with lines (i.e., density>0), border = TRUE uses the same color for the border as for the shading lines. Otherwise, fill color or shaded line color are specified in col while border color is specified in border.

col — A vector of bar fill colors or line colors when bars are drawn and density != 0, repeated as necessary to exceed the number of bars. Also used for the bar borders when border = TRUE.

col.dfunc — Color of the distance function(s). If only one distance function (one line) is being plotted, the default color is "red". If covariates or newdata are present, the default value uses graphics::rainbow(n), where n is the number of plotted distance functions. Otherwise, col.dfunc is replicated to the required length. Plot all distance functions in the same color by setting col.dfunc to a scalar. Plot blue-red pairs of distance functions by setting col.dfunc = c("blue", "red"). Etc.

lty.dfunc — Line type of the distance function(s). If covariates or newdata is present, the default uses line types to 1:n, where n is the number of plotted distance functions. Otherwise, lty.dfunc is replicated to the required length. Plot solid lines by specifying lty.dfunc = 1. Plot solid-dashed line pairs by specifying lty.dfunc = c(1,2). Etc.

lwd.dfunc — Line width of the distance function(s), replicated to the required length. Default is 2 for all lines.

... — When bars are plotted, this routine uses graphics::barplot to draw the plotting region and bars. When bars are not plotted, this routine sets up the plot with graphics::plot. ...can be any argument to barplot or plot EXCEPT width, ylim, xlim, density, angle, and space. For example, set the main title with main = "Main Title".

Value

The input distance function is returned, with two additional components than can be used to reconstruct the plotted bars. (To obtain values of the plotted distance functions, use `predict` with `type = "distances"`.) The additional components are:

`barHeights`	A vector containing the scaled bar heights drawn on the plot.
`barWidths`	A vector or scalar of the bar widths drawn on the plot, with measurement units.

Re-plot the bars with `barplot( return$barHeights, width = return$barWidths )`.

See Also

`plot.dfunc`

Examples

```
# Example data
set.seed(87654)
x <- rnorm(1000, mean=0, sd=20)
x <- x[x >= 0]
x <- units::set_units(x, "ft")
Df <- data.frame(transectID = "A"
                , distance = x
                 ) |>
  dplyr::nest_by( transectID
                , .key = "detections") |>
  dplyr::mutate(length = units::set_units(1,"mi"))
attr(Df, "detectionColumn") <- "detections"
attr(Df, "obsType") <- "single"
attr(Df, "transType") <- "line"
attr(Df, "effortColumn") <- "length"
is.RdistDf(Df) # TRUE

# Estimation
dfunc <- dfuncEstim(Df
                  , formula = distance~1
                  , likelihood="halfnorm")
plot(dfunc)
plot(dfunc, nbins=25)
```

`predDensity` *predDensity - Density on transects*

Description

An internal prediction method for computing density on the sampled transects.

Usage

```
predDensity(object, propUnitSurveyed = 1)
```

Arguments

object	An Rdistance model frame or fitted distance function, normally produced by a call to dfuncEstim.
propUnitSurveyed	A scalar or vector of real numbers between 0 and 1. The proportion of the default sampling unit that was surveyed. If both sides of line transects were observed, propUnitSurveyed = 1. If only a single side of line transects were observed, set propUnitSurveyed = 0.5. For point transects, this should be set to the proportion of each circle that was observed. Length must either be 1 or the total number of transects in x.

Value

A data frame containing the original data used to fit the distance function, plus an additional column containing the density of individuals on each transect.

Examples

```
data(sparrowDfuncObserver)
predict(sparrowDfuncObserver, type="density")
```

---

predDfuncs *predDfuncs - Predict distance functions*

Description

An internal prediction function to predict a distance function. This version allows for matrix inputs and uses matrix operations, and is thus faster than earlier looping versions.

Usage

```
predDfuncs(object, params, distances, isSmooth)
```

Arguments

object	An Rdistance model frame or fitted distance function, normally produced by a call to dfuncEstim.
params	A matrix of distance function parameters. Rows are observations, columns contain the set of parameters (canonical and expansion) for each observation.

distances	A vector or 1-column matrix of distances at which to evaluate distance functions, when distance functions are requested. distances must have measurement units. Any distances outside the observation strip (object$w.lo to object$w.hi) are discarded. If distances is NULL, a sequence of getOption("Rdistance_intEvalPts") (default 101) evenly spaced distances between object$w.lo and object$w.hi (inclusive) is used.
isSmooth	Logical. TRUE if the distance function is smoothed (and hence has no parameters).

Value

A matrix of distance function values, of size length(distances) X nrow(params). Each row of params is associated with a column, i.e., a different distance function. Distances are associated with rows, i.e., use matplot(d,return) to plot values on separate distance functions specified by rows of params.

---

predict.dfunc *predict.dfunc - Predict distance functions*

Description

Predict either likelihood parameters, distance functions, site-specific density, or site-specific abundance from estimated distance function objects.

Usage

```
## S3 method for class 'dfunc'
predict(
  object,
  newdata = NULL,
  type = c("parameters"),
  distances = NULL,
  propUnitSurveyed = 1,
  area = NULL,
  ...
)
```

Arguments

object	An Rdistance model frame or fitted distance function, normally produced by a call to dfuncEstim.
newdata	A data frame containing new values of the covariates at which to evaluate the distance functions. If newdata is NULL, distance functions are evaluated at values of the observed covariates and results in one prediction per distance or transect (see parameter type). If newdata is not NULL and the model does not contains covariates, this routine returns one prediction for each row in newdata, but columns and values in newdata are ignored.
type	The type of predictions desired.

- **If** `type` **== "parameters"**: Returned value is a matrix of predicted (canonical) parameters of the likelihood function. If `newdata` is NULL, return contains one parameter value for every detection distance in `object$mf` (distances in `object$mf` are between `object$w.lo` and `object$w.hi` and non-missing). If `newdata` is not NULL, returned vector has one parameter for every row in `newdata`. Parameter `distances` is ignored when `type` == "parameters". Canonical parameters (non-expansion terms) are returned on the response (inverse-link) scale. Raw canonical parameters in `object$par` are stored in the link scale. Expansion term parameters use the identity link, so their value in the output equals their value in `object$par`.
- **If** `type` **== "likelihood"**: Returned value is a matrix of **unscaled** likelihood values for all observed distances in `object$mf`, i.e., raw distance functions evaluated at the observed distances. Parameters `newdata` and `distances` are ignored when `type` is "likelihood". The negative log likelihood of the full data set is `-sum(log(predict(object,type="likelihood") / effectiveDistance(object)))`.
- **If** `type` **== "dfuncs" or "dfunc"**: Returned value is a matrix whose columns contain scaled distance functions. The distance functions in each column are evaluated at distances in argument `distances`, not at the observed distances in `object$mf`. The number of distance functions returned (i.e., number of columns) depends on `newdata` as follows:
  - If `newdata` is NULL, one distance function will be returned for every detection in `object$mf` that has valid covariate values.
  - If `newdata` is not NULL, one distance function will be returned for each observation (row) in `newdata`.
- **If** `type` **== "density" or "abundance"**: Returned object is a tibble containing predicted density and abundance on the area surveyed by each transect.

If `object` is a smoothed distance function, it does not have parameters and this routine will only return scaled distance functions, densities, or abundances. That is, `type` = "parameters" when `object` is smoothed does not make sense and the smoothed distance function estimate will be returned if `type` does not equal "density" or "abundance".

`distances` A vector or 1-column matrix of distances at which to evaluate distance functions, when distance functions are requested. `distances` must have measurement units. Any distances outside the observation strip (`object$w.lo` to `object$w.hi`) are discarded. If `distances` is NULL, a sequence of `getOption("Rdistance_intEvalPts")` (default 101) evenly spaced distances between `object$w.lo` and `object$w.hi` (inclusive) is used.

`propUnitSurveyed`

A scalar or vector of real numbers between 0 and 1. The proportion of the default sampling unit that was surveyed. If both sides of line transects were observed, `propUnitSurveyed` = 1. If only a single side of line transects were observed, set `propUnitSurveyed` = 0.5. For point transects, this should be set to the proportion of each circle that was observed. Length must either be 1 or the total number of transects in `x`.

area — A scalar containing the total area of inference. Usually, this is study area size. If area is NULL (the default), area will be set to 1 square unit of the output units and density estimates will be produced. If area is not NULL, it must have measurement units assigned by the units package. The units on area must be convertible to squared output units. Units on area must be two-dimensional. For example, if output units are "foo", units on area must be convertible to "foo^2" by the units package. Units of "km^2", "cm^2", "ha", "m^2", "acre", "mi^2", and several others are acceptable.

... — Included for compatibility with generic predict methods.

Value

A matrix containing predictions:

- **If** type **is "parameters"**, the returned matrix contains likelihood parameters. The extent of the first dimension (rows) in the returned matrix is equal to either the number of detection distances in the observed strip or number of rows in newdata. The returned matrix's second dimension (columns) is the number of parameters in the likelihood plus the number of expansion terms. See the help for each likelihoods to interpret returned parameter values. All parameters are returned on the inverse-link scale; i.e., *exponential* for canonical parameters and *identity* for expansion terms.
- **If** type **is "dfuncs" or "dfunc"**, columns of the returned matrix contains detection functions (i.e., *g(x)*). The extent of the first dimension (number of rows) is either the number of distances specified in distances or options()$Rdistance_intEvalPts if distances is not specified. The extent of the second dimension (number of columns) is:
  - the number of detections with non-missing distances: if newdata is NULL.
  - the number of rows in newdata if newdata is specified.

  All distance functions in columns of the return are scaled to object$g.x.scale at object$x.scl. The returned matrix has the following additional attributes:
  - attr(return, "distances") is the vector of distances used to predict the function in return. Either the input distances object or the computed sequence of distances when distances is NULL.
  - attr(return, "x0") is the vector of distances at which each distance function in return was scaled. i.e., the vector of x.scl.
  - attr(return, "g.x.scl") is the height of *g(x)* (the distance function) at *x0*.
- **If** type **is "density" or "abundance"**, the return is a tibble containing density and abundance estimates by transect. All transects in the input data (i.e., object$data) are included, even those with missing lengths. Columns in the tibble are:
  - transect ID: the grouping factor of the original RdistDf object.
  - individualsSeen: sum of non-missing group sizes on that transect.
  - avgPdetect: average probability of detection over groups sighted on that transect.
  - effort: size of the area surveyed by that transect.
  - density: density of individuals in the area surveyed by the transect.
  - abundance: abundance of individuals in the area surveyed by the transect.

See Also

halfnorm.like, negexp.like, hazrate.like

Examples

```
data("sparrowDf")

# For dimension checks:
nd <- getOption("Rdistance_intEvalPts")

# No covariates
dfuncObs <- sparrowDf |> dfuncEstim(formula = dist ~ 1
                     , w.hi = units::as_units(100, "m"))

n  <- nrow(dfuncObs$mf)
p <- predict(dfuncObs) # parameters
all(dim(p) == c(n, 1))

# values in newdata ignored because no covariates
p <- predict(dfuncObs, newdata = data.frame(x = 1:5))
all(dim(p) == c(5, 1))

# Distance functions in columns, one per observation
p <- predict(dfuncObs, type = "dfunc")
all(dim(p) == c(nd, n))

d <- units::set_units(c(0, 20, 40), "ft")
p <- predict(dfuncObs, distances = d, type = "dfunc")
all(dim(p) == c(3, n))

p <- predict(dfuncObs
   , newdata = data.frame(x = 1:5)
   , distances = d
   , type = "dfunc")
all(dim(p) == c(3, 5))

# Covariates
data(sparrowDfuncObserver) # pre-estimated object
## Not run:
# Command to generate 'sparrowDfuncObserver'
sparrowDfuncObserver <- sparrowDf |>
            dfuncEstim(formula = dist ~ observer
                     , likelihood = "hazrate")

## End(Not run)

predict(sparrowDfuncObserver)  # n X 2

Observers <- data.frame(observer = levels(sparrowDf$observer))
predict(sparrowDfuncObserver, newdata = Observers) # 5 X 2

predict(sparrowDfuncObserver, type = "dfunc") # nd X n
```

```
predict(sparrowDfuncObserver, newdata = Observers, type = "dfunc") # nd X 5
d <- units::set_units(c(0, 150, 400), "ft")
predict(sparrowDfuncObserver
  , newdata = Observers
  , distances = d
  , type = "dfunc") # 3 X 5

# Density and abundance by transect
predict(sparrowDfuncObserver
  , type = "density")
```

---

predLikelihood *predLikelihood - Distance function values at observations*

Description

An internal prediction function to predict (compute) the values of distance functions at a set of observed values. Unlike `predDfuncs`, which evaluates distance functions at EVERY input distance, this routine evaluates distance functions at only ONE distance. This is what's appropriate for likelihood computation. This version allows for matrix inputs and uses matrix operations, and is thus faster than earlier looping versions.

Usage

```
predLikelihood(object, params)
```

Arguments

`object`	An Rdistance model frame or fitted distance function, normally produced by a call to `dfuncEstim`.
`params`	A matrix of distance function parameters. Rows are observations, columns contain the set of parameters (canonical and expansion) for each observation.

Details

Assuming `L` is the vector returned by this function, the negative log likelihood is `-sum(log(L / I), na.rm=T)`, where `I` is the integration constant, or area under the likelihood between `w.lo` and `w.hi`. Note that returned likelihood values for distances less than `w.lo` or greater than `w.hi` are `NA`; hence, `na.rm=TRUE` in the sum.

Value

A vector of distance function values, of length n = number of observed distances = length(distances(x)). Elements in `distances(x)` correspond, in order, to values in the returned vector.

print.abund *Print abundance estimates*

Description

Print an object of class c("abund","dfunc") produced by abundEstim.

Usage

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

Arguments

x	An object output by abundEstim. This is a distance function object augmented with abundance estimates, and has class c("abund", "dfunc").
...	Included for compatibility to other print methods. Ignored here.

Value

0 is invisibly returned

See Also

dfuncEstim, abundEstim, summary.dfunc, print.dfunc, summary.abund

Examples

```
# Load example sparrow data (line transect survey type)
data(sparrowDf)

# Fit half-normal detection function
dfunc <- sparrowDf |> dfuncEstim(formula=dist~groupsize(groupsize))

# Estimate abundance given a detection function
fit <- abundEstim(object = dfunc
                , area = units::set_units(4105, "km^2")
                , ci = NULL)
print(fit)
summary(fit)

## Not run:
# Bootstrap confidence intervals (500 iterations)
# Requires ~4 min
fit <- abundEstim(object = dfunc
                , area = units::set_units(4105, "km^2")
                , ci = 0.95
                , plot.bs = TRUE
                , showProgress = TRUE)
```

```
print(fit)
summary(fit)

## End(Not run)
```

`print.dfunc` *print.dfunc - Print method for distance function object*

Description

Print method for distance function objects produced by `dfuncEstim`.

Usage

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

Arguments

`x`	An estimated detection function object, normally produced by calling `dfuncEstim`.
`...`	Included for compatibility with other print methods. Ignored here.

Value

The input distance function (`x`) is returned invisibly.

See Also

`dfuncEstim`, `plot.dfunc`, `print.abund`, `summary.dfunc`

Examples

```
# Load example sparrow data (line transect survey type)
data(sparrowSiteData)
data(sparrowDetectionData)

# Fit half-normal detection function
sparrowDf <- RdistDf(sparrowSiteData, sparrowDetectionData)
dfunc <- sparrowDf |> dfuncEstim(formula=dist~1)

dfunc
```

RdistanceControls *Rdistance optimization control parameters.*

Description

Optimization control parameters are set by calls to `options()` (see examples). Optimization parameters used in `Rdistance` are the following:

- `Rdist_maxIters`: The maximum number of optimization iterations allowed.
- `Rdist_evalMax`: The maximum number of objective function evaluations allowed.
- `Rdist_likeTol`: Minimum change in the likelihood between iterations required optimization to continue. If the likelihood changes by less than this amount, optimization stops and a solution is declared. Iteration continues when likelihood changes exceed this value.
- `Rdist_coefTol`: Minimum change in model coefficients between iterations for optimization to continue. If the sum of squared coefficient differences changes by less than this amount between iterations, optimization stops and a solution is declared.
- `Rdist_optimizer`: A string specifying the optimizer to use. Results can vary between optimizers, so switching algorithms sometimes makes a poorly behaved distance function converge. Valid values are "optim" which uses `optim::optim`, and "nlminb" which uses `stats:nlminb`. The authors have had better luck with "nlminb" than "optim" and "nlminb" runs noticeably faster. Problems with solutions near, but not on, parameter boundaries may require use of "optim".
- `Rdist_hessEps`: A vector of parameter distances used during computation of numeric second derivatives. These distances control and determine variance estimates, and they may need revision when the maximum likelihood solution is near a parameter boundary. Should have length 1 or the number of parameters in the model. See function `secondDeriv` for further details.
- `Rdist_requireUnits`: A logical specifying whether measurement units are required on distances and areas. If TRUE, measurement units are required on off-transect and radial distances in the input data frame. Likewise, measurement units are required on truncation distances, scale location, transect lengths, and study area size. If FALSE, no units are required and input values are used as is. The FALSE options is provided for rare cases when `Rdistance` functions are called from other functions and the calling functions do not accommodate units.

  Assign units with statement like `units(detectionDf$dist) <- "m"` or `units::set_units(w.hi, "km")`. Measurement units of the various physical quantities need not be equal because appropriate conversions occur internally. An error is thrown if differing units are not compatible. For example, "m" (meters) cannot be converted into "ha" (hectares), but "acres" can be converted into "ha". `Rdistance` recognizes units listed in `units::valid_udunits()`.
- `Rdist_maxBSFailPropForWarning`: The proportion of bootstrap iterations that can fail without a warning. If the proportion of non-convergent bootstrap iterations exceeds this parameter, a warning about the validity of CI's is issued in the abundance print method.

Examples

```
# increase number of iterations
options(Rdist_maxIters=2000)

# change optimizer and decrease tolerance
options(list(Rdist_optimizer="optim", Rdist_likeTol=1e-6))
```

RdistDf *RdistDf - Construct Rdistance nested data frames*

Description

Makes an Rdistance data frame from separate transect and detection data frames. Rdistance data frames are nested data frames with one row per transect. Detection information for each transect appears in a list-based column that itself contains a data frame. See **Rdistance Data Frames**.

Rdistance data frames can be constructed using calls to dplyr::nest_by and dplyr::right_jion, with subsequent attribute assignment (see **Examples**). This routine is a convenient wrapper for those calls.

Usage

```
RdistDf(
  transectDf,
  detectionDf,
  by = NULL,
  pointSurvey = FALSE,
  observer = "single",
  .detectionCol = "detections",
  .effortCol = NULL
)
```

Arguments

transectDf	A data frame with one row per transect and columns containing information about the entire transect. At a minimum, this data frame must contain the transect's ID so it can be merged with detectionDf, (see parameter by) and the amount of effort the transect represents (see parameter .effortCol). All detections are made on a transect, but not all transects require detections. That is, transectDf should contain rows, and hence ID's and lengths, of all surveyed transects, even those on which no targets were detected (so-called "zero transects"). Transect-level covariates, such as habitat type, elevation, or observer IDs, should appear as variables in this data frame.
detectionDf	A data frame containing detection information associated with each transect. At a minimum, each row of this data frame must contain the following:

- **Transect IDs**: The ID of the transect on which a target group was detected so that the detection data frame can be merged with `transectDf` (see parameter by). Multiple detections on the same transect are possible and hence multiple rows in `detectonDf` can contain the same transect ID.
- **Detection Distances**: The distance at which each detection was made. The distance column will eventually be specified on the left-hand side of `formula` in a call to `dfuncEstim`. As of Rdistance version 3.0.0, detection distances must have physical measurement units assigned. See **Measurement Units**.

Optional columns in 'detectionDf':

- **Group sizes**:If sighted targets vary in size, or group sizes are not all 1, `detectionDf` must also contain a column specifying group sizes. Non-unity group sizes are specified using `+groupsize(columnName)` on the right-hand-side of `formula` in an eventual call to `dfuncEstim`.
- **Detection Level Covariates**: Such as sex, color, habitat, etc.

`by` A character vector of variables to use in the join. The right-hand side of this join identifies unique transects (unique rows) in both `transectDf` and the output (see warning in **Details**). If NULL, the join will be 'natural', using all common variables in `transectDf` and `detectionDf`. To join on specific variables, specify a character vector of the variables. For example, by = c("a", "b") joins `transectDf$a` to `detectionDf$a` and `transectDf$b` to `detectionDf$b`. If join variable names differ between `transectDf` and `detectionDf`, use a named character vector like by = c("a" = "b", "c" = "d") which joins `transectDf$a` to `detectionDf$b` and `transectDf$c` to `detectionDf$d`.

`pointSurvey` If TRUE, observations were made from discrete points (i.e., during a point-transect survey) and distances are radial from observation point to target. If FALSE, observations were made along a continuous transect (i.e., during a line-transect survey) and distances are from target to nearest point on the transect (i.e., perpendicular to transect).

`observer` Type of observer system. Legal values are **"single"** for single observer systems, **"1given2"** for a double observer system wherein observations made by observer 1 are tested against observations made by observer 2, **"2given1"** for a double observer system wherein observations made by observer 2 are tested against observations made by observer 1, and **"both"** for a double observer system wherein observations made by both observers are tested against the other and combined.

`.detectionCol` Name of the list column that will contain detection data frames. Default name is "detections". Detection distances (LHS of 'dfuncEstim' formula) and group sizes are normally columns in the nested detection data frames embedded in '.detectionCol'.

`.effortCol` For continuous line transects, this specifies the name of a column in `transectDf` containing transect lengths, which must have measurement units. For point transects, this specifies the name of a column containing the number of points on each transect. The effort column for point transects *cannot* contain measurement units. Default is "length" for line-transects, "numPoints" for point-transects. If those names are not found, the first column in the merged data frame whose

name contains 'point' (for point transects) or 'length' (for line transects) is used and a message is printed. Matching is case insensitive, so for example, 'nPoints' and 'N_point' and 'numberOfPoints' will all be matched. If two or more column names match the effort column search terms, a warning is issued. See **Transect Lengths** for a description of point and line transects.

Details

For valid bootstrap estimates of confidence intervals (computed in `abundEstim`), each row of the nested data frame must represent one transect (more generally, one sampling unit), and none should be duplicated. The combination of transect columns in `by` (i.e., the LHS of the merge, or "a" and "b" of `by = c("a" = "d", "b" = "c")` for example) should specify *unique* transects and unique rows of `transectDf`. **Warning:** If `by` does not specify unique rows of `transectDf`, `dplyr::left_join`, which is called internally, will perform a many-to-many merge without warning, and this normally duplicates both transects and detections.

Value

A nested tibble (a generalization of base data frames) with one row per transect, and detection information in a list column. Technically, the return is a grouped `tibble` from the `tibble` package with one row per group, and a list column containing detection information. Survey type, observer system, and name of the effort column are recorded as attributes (`transType`, `obsType`, and `effortColumn`, respectfully). The return prints nicely using methods in package `tibble`. If returned objects print strangely, attach library `tibble`. A summary method tailored to distance sampling is available (i.e., `summary(return)`).

Rdistance Data Frames

`RdistDf` data frames contain the following information:

- **Transect Information**: Each row of the data frame contains transect id and effort. Effort is transect length for line-transects, and number of points for point-transects. Optionally, transect-level covariates (such as habitat or observer id) appear on each row.
- **Detection Information**: Observation distances (either perpendicular off-transect or radial off-point) appear in a data frame stored in a list column. If detected groups occasionally included more than one target, a group size column must be present in the list-column data frame. Optionally, detection-level covariates (such as sex or size) can appear in the data frame of the list column.
- **Distance Type**: The type of observation distances, either perpendicular off-transect (for line-transects studies) or radial off-point (for point-transect studies) must appear as an attribute of `RdistDf`'s.
- **Observer Type**: The type of observation system used, either single observer or one of three types of multiple observer systems, must appear as an attribute of `RdistDf`'s.

Transect Lengths

Line-transects are continuous paths with targets detectable at any point. Point transects consist of one or more discrete points along a path from which observers search for targets. The length of a line-transect is it's physical length (e.g., km or miles). The 'length' of a point transect is the number

of points along the transect. Single points are considered transects of length one. The length of line-transects must have a physical measurement unit (e.g., 'm' or 'ft'). The length of point-transects must be a unit-less integers (i.e., number of points).

Measurement Units

As of `Rdistance` version 3.0.0, measurement units are require on all physical distances. Requiring units ensures that internal calculations and results (e.g., ESW and abundance) are correct and that output units are clear. Physical distances are required on off-transect distances, radial distances, truncation distances (`w.lo`, unless it is zero; and `w.hi`, unless it is NULL), scale locations (`x.scl`, unless it is zero), line-transect lengths, and study area size. All units are 1-dimensional except those on study area, which are 2-dimensional.

Physical measurement units can vary. For example, off-transect distances can be meters ("m"), `w.hi` can be inches ("in"), and `w.lo` can be kilometers ("km"). Internally, all distances are converted to the units specified by `outputUnits` (or the units of input distances if `outputUnits` is NULL), and all output is reported in units of `outputUnits`. Valid conversions must exist between units or an error is thrown. For example, meters cannot be converted into hectares.

Measurement units can be assigned using `units()<-` after attaching the `units` package or with `x <- units::set_units(x, "<units>")`. See `units::valid_udunits()` for a list of valid symbolic units.

If measurements are truly unit-less, or measurement units are unknown, set `options(Rdist_requireUnits = FALSE)`. This suppresses all unit checks and conversions. Users are on their own to make sure inputs are scaled correctly and that output units are known.

See Also

`is.RdistDf`: check validity of RdistDf data frames; `dfuncEstim`: estimate distance function.

Examples

```
data(sparrowSiteData)
data(sparrowDetectionData)

sparrowDf <- RdistDf( sparrowSiteData, sparrowDetectionData )
is.RdistDf(sparrowDf, verbose = T)
summary(sparrowDf)
summary(sparrowDf
      , formula = dist ~ groupsize(groupsize)
      , w.hi = units::set_units(100, "m"))

# Equivalent to above:
sparrowDf <- sparrowDetectionData |>
  dplyr::nest_by( siteID
                , .key = "detections") |>
  dplyr::right_join(sparrowSiteData, by = "siteID")
attr(sparrowDf, "detectionColumn") <- "detections"
attr(sparrowDf, "effortColumn") <- "length"
attr(sparrowDf, "obsType") <- "single"
attr(sparrowDf, "transType") <- "line"
is.RdistDf(sparrowDf, verbose = T)
```

```
summary(sparrowDf, formula = dist ~ groupsize(groupsize))

# Condensed view: 1 row per transect (make sure tibble is attached)
sparrowDf

# Expansion methods:
# (1) use Rdistance::unnest (includes zero transects)
df1 <- unnest(sparrowDf)
any( df1$siteID == "B2" )  # TRUE

# Use tidyr::unnest(); but, no zero transects
df2 <- tidyr::unnest(sparrowDf, cols = "detections")
any( df2$siteID == "B2" )  # FALSE

# Use dplyr::reframe for specific transects (e.g., for transect "B3")
sparrowDf |>
  dplyr::filter(siteID == "B3") |>
  dplyr::reframe(detections)

# Count detections per transect (can't use dplyr::if_else)
df3 <- sparrowDf |>
  dplyr::reframe(nDetections = ifelse(is.null(detections), 0, nrow(detections)))
sum(df3$nDetections) # Number of detections
sum(df3$nDetections == 0) # Number of zero transects

# Point transects
data(thrasherDetectionData)
data(thrasherSiteData)
thrasherDf <- RdistDf( thrasherSiteData
               , thrasherDetectionData
               , pointSurvey = TRUE
               , by = "siteID"
               , .detectionCol = "detections")
summary(thrasherDf, formula = dist ~ groupsize(groupsize))
```

`secondDeriv` *Numeric second derivatives*

Description

Computes numeric second derivatives (hessian) of an arbitrary multidimensional function at a particular location.

Usage

```
secondDeriv(x, FUN, eps = 1e-08, ...)
```

Arguments

x	The location (a vector) where the second derivatives of FUN are desired.
FUN	An R function for which the second derivatives are sought. This must be a function of the form FUN <- function(x, ...){...} where x is a vector of variable parameters to FUN at which to evaluate the 2nd derivative, and ... are additional parameters needed to evaluate the function. FUN must return a single value (scalar), the height of the surface above x, i.e., FUN evaluated at x.
eps	A vector of small relative distances to add to x when evaluating derivatives. This determines the 'dx' of the numerical derivatives. That is, the function is evaluated at x, x+dx, and x+2*dx, where dx = x*eps^0.25, in order to compute the second derivative. eps defaults to 1e-8 for all dimensions which equates to setting dx to one percent of each x (i.e., by default the function is evaluate at x, 1.01*x and 1.02*x to compute the second derivative). One might want to change eps if the scale of dimensions in x varies wildly (e.g., kilometers and millimeters), or if changes between FUN(x) and FUN(x*1.01) are below machine precision. If length of eps is less than length of x, eps is replicated to the length of x.
...	Any arguments passed to FUN.

Details

This function uses the "5-point" numeric second derivative method advocated in numerous numerical recipe texts. During computation of the 2nd derivative, FUN must be capable of being evaluated at numerous locations within a hyper-ellipsoid with cardinal radii 2*x*(eps)^0.25 = 0.02*x at the default value of eps.

A handy way to use this function is to call an optimization routine like nlminb with FUN, then call this function with the optimized values (solution) and FUN. This will yield the hessian at the solution and this is can produce a better estimate of the variance-covariance matrix than using the hessian returned by some optimization routines. Some optimization routines return the hessian evaluated at the next-to-last step of optimization.

An estimate of the variance-covariance matrix, which is used in Rdistance, is solve(hessian) where hessian is secondDeriv(<parameter estimates>, <likelihood>).

Examples

```
func <- function(x){-x*x} # second derivative should be -2
secondDeriv(0,func)
secondDeriv(3,func)

func <- function(x){3 + 5*x^2 + 2*x^3} # second derivative should be 10+12x
secondDeriv(0,func)
secondDeriv(2,func)

func <- function(x){x[1]^2 + 5*x[2]^2} # should be rbind(c(2,0),c(0,10))
secondDeriv(c(1,1),func)
```

`simple.expansion` *Calculate simple polynomial expansion for detection function likelihoods*

Description

Computes simple polynomial expansion terms used in the likelihood of a distance analysis. More generally, will compute polynomial expansions of any numeric vector.

Usage

```
simple.expansion(x, expansions)
```

Arguments

`x`	In a distance analysis, `x` is a numeric vector of the proportion of a strip transect's half-width at which a group of individuals were sighted. If w is the strip transect half-width or maximum sighting distance, and d is the perpendicular off-transect distance to a sighted group ($d \leq w$), `x` is usually d/w. More generally, `x` is a vector of numeric values
`expansions`	A scalar specifying the number of expansion terms to compute. Must be one of the integers 1, 2, 3, or 4.

Details

The polynomials computed here are:

- **First term**:
$$h_1(x) = x^4,$$
- **Second term**:
$$h_2(x) = x^6,$$
- **Third term**:
$$h_3(x) = x^8,$$
- **Fourth term**:
$$h_4(x) = x^{10},$$

The maximum number of expansion terms computed is 4.

Value

A matrix of size `length(x)` X `expansions`. The columns of this matrix are the Simple polynomial expansions of `x`. Column 1 is the first expansion term of `x`, column 2 is the second `x`, and so on up to `expansions`.

See Also

`dfuncEstim`, `cosine.expansion`, `hermite.expansion`

Examples

```
x <- seq(0, 1, length = 200)
simp.expn <- simple.expansion(x, 4)
plot(range(x), range(simp.expn), type="n")
matlines(x, simp.expn, col=rainbow(4), lty = 1)
```

`sparrowDetectionData` *Brewer's Sparrow detection data*

Description

Detection data from line transect surveys for Brewer's sparrow on 72 transects located on a 4105 km^2 study area in central Wyoming. Data were collected by Dr. Jason Carlisle of the Wyoming Cooperative Fish & Wildlife Research Unit in 2012. Each transect was 500 meters long.

Format

A data.frame containing 356 rows and 5 columns. Each row represents a detected group of sparrows. Column descriptions:

1. `siteID`: Factor (72 levels), the site or transect where the detection was made.
2. `groupsize`: Number, the number of individuals within the detected group.
3. `sightdist`: Number, distance (m) from the observer to the detected group.
4. `sightangle`: Number, the angle (degrees) from the transect line to the detected group.
5. `dist`: Number, the perpendicular, off-transect distance (m) from the transect to the detected group. This is the distance used in analysis. Calculated using `perpDists`.

Source

The Brewer's sparrow data are a subset of the data collected by Jason Carlisle and various field technicians for his Ph.D. from the Department of Ecology, University of Wyoming, in 2017. This portion of Jason's work was funded by the Wyoming Game and Fish Department through agreements with the University of Wyoming's Cooperative Fish & Wildlife Research Unit (2012).

References

Carlisle, J.D. 2017. The effect of sage-grouse conservation on wildlife species of concern: implications for the umbrella species concept. Dissertation. University of Wyoming, Laramie, Wyoming, USA.

Carlisle, J. D., and A. D. Chalfoun. 2020. The abundance of Greater Sage-Grouse as a proxy for the abundance of sagebrush-associated songbirds in Wyoming, USA. *Avian Conservation and Ecology* 15(2):16. doi:10.5751/ACE01702150216

See Also

`sparrowSiteData`

`sparrowDf` *Brewer's Sparrow detection data frame in Rdistance >4.0.0 format.*

Description

Detection data from line transect surveys for Brewer's sparrow on 72 transects located on a 4105 km^2 study area in central Wyoming collected by Dr. Jason Carlisle as part of his graduate work in the Wyoming Cooperative Fish & Wildlife Research Unit in 2012. Each transect was 500 meters long.

Format

A rowwise tibble containing 72 rows and 9 columns, one of which is nested data frame of detections. Each row represents one transect. The embedded data frame in column `detections` contains the detections made on the transect represented on that row.

Column descriptions:

1. `siteID`: Factor (72 levels), the transect identifier for that row of the data frame.
2. `length`: The length, in meters [m], of each transect.
3. `observer`: Identity of the observer who surveyed the transect.
4. `bare`: The mean bare ground cover (%) within 100 [m] of the transect.
5. `herb`: The mean herbaceous cover (%) within 100 [m] of the transect.
6. `shrub`: The mean shrub cover (%) within 100 [m] of the transect.
7. `height`: The mean shrub height [cm] within 100 [m] of the transect.
8. `shrubclass`: Shrub class factor. Either "Low"" when shrub cover is < 10%, or "High" if cover >= 10%.

The embedded data frame in column `detections` contains the following variables:

1. `groupsize`: The number of individuals in the detected group.
2. `sightdist`: Distance [m] from observer to the detected group.
3. `sightangle`: Angle [degrees] from the transect line to the detected group. Not bearing. Range 0 [degrees] to 90 [degrees].
4. `dist`: Perpendicular, off-transect distance [m], from the transect to the detected group. This is the distance used in analysis. Calculated using `perpDists`.

Source

The Brewer's sparrow data are a subset of data collected by Jason Carlisle and various field technicians for his Ph.D. from the Department of Ecology, University of Wyoming, in 2017. This portion of Jason's work was funded by the Wyoming Game and Fish Department through agreements with the University of Wyoming's Cooperative Fish & Wildlife Research Unit (2012).

References

Carlisle, J.D. 2017. The effect of sage-grouse conservation on wildlife species of concern: implications for the umbrella species concept. Dissertation. University of Wyoming, Laramie, Wyoming, USA.

Carlisle, J. D., and A. D. Chalfoun. 2020. The abundance of Greater Sage-Grouse as a proxy for the abundance of sagebrush-associated songbirds in Wyoming, USA. *Avian Conservation and Ecology* 15(2):16. doi:10.5751/ACE01702150216

See Also

sparrowSiteData, sparrowDetectionData, RdistDf

Examples

```
## Not run:
# The following code generated 'sparrowDf'
data(sparrowDetectionData)
data(sparrowSiteData)
sparrowDf <- RdistDf(transectDf = sparrowSiteData
                   , detectionDf = sparrowDetectionData
                   , by = "siteID"
                   , pointSurvey = FALSE
                   , .effortCol = "length"
                    )

## End(Not run)

data(sparrowDf)
tidyr::unnest(sparrowDf, detections)  # only non-zero transects
Rdistance::unnest(sparrowDf) # with zero transects at the bottom
summary(sparrowDf,
  formula = dist ~ groupsize(groupsize)
)
```

---

sparrowDfuncObserver *Brewer's Sparrow detection function*

Description

Pre-estimated Brewer's sparrow detection function that included and 'observer' effect. Included to speed up example execution times. See 'Examples'.

Format

An estimated distance function object with class 'dfunc'. See 'Value' section of dfuncEstim for description of components.

See Also

sparrowSiteData and sparrowDetectionData for description of the data

Examples

```
## Not run:
# the following code was used to generate 'sparrowDfuncObserver'
data(sparrowDf)
sparrowDfuncObserver <- sparrowDf |>
            dfuncEstim(formula = dist ~ observer
                     , likelihood = "hazrate")

## End(Not run)
```

---

sparrowSiteData *Brewer's Sparrow site data*

Description

Site data from line transect surveys for Brewer's sparrow on 72 transects located on a 4105 km^2 study area in central Wyoming. Data were collected by Dr. Jason Carlisle of the Wyoming Cooperative Fish & Wildlife Research Unit in 2012. Each transect was 500 meters long.

Format

A data.frame containing 72 rows and 8 columns. Each row represents a site (transect) surveyed. Column descriptions:

1. siteID: Factor (72 levels), the site or transect surveyed.
2. length: Number, the length (m) of each transect.
3. observer: Factor (five levels), identity of the observer who surveyed the transect.
4. bare: Number, the mean bare ground cover (%) within 100 m of each transect.
5. herb: Number, the mean herbaceous cover (%) within 100 m of each transect.
6. shrub: Number, the mean shrub cover (%) within 100 m of each transect.
7. height: Number, the mean shrub height (cm) within 100 m of each transect.
8. shrubclass: Factor (two levels), shrub class is "Low"" when shrub cover is < 10%, "High" otherwise.

Source

The Brewer's sparrow data are a subset of the data collected by Jason Carlisle and various field technicians for his Ph.D. from the Department of Ecology, University of Wyoming, in 2017. This portion of Jason's work was funded by the Wyoming Game and Fish Department through agreements with the University of Wyoming's Cooperative Fish & Wildlife Research Unit (2012).

References

Carlisle, J.D. 2017. The effect of sage-grouse conservation on wildlife species of concern: Implications for the umbrella species concept. Dissertation. University of Wyoming, Laramie, Wyoming, USA.

Carlisle, J. D., and A. D. Chalfoun. 2020. The abundance of Greater Sage-Grouse as a proxy for the abundance of sagebrush-associated songbirds in Wyoming, USA. *Avian Conservation and Ecology* 15(2):16. doi:10.5751/ACE01702150216

See Also

sparrowDetectionData

`startLimits` *startLimits - Distance function starting values and limits*

Description

Returns starting values and limits (boundaries) for the parameters of distance functions. This function is called by other routines in `Rdistance`, and is not intended to be called by the user. Replace this function in the global environment to change boundaries and starting values.

Usage

```
startLimits(ml)
```

Arguments

`ml`	Either a Rdistance 'model frame' or an Rdistance 'fitted object'. Both are of class "dfunc". Rdistance 'model frames' are lists containing components necessary to estimate a distance function, but no estimates. Rdistance 'model frames' are typically produced by calls to `parseModel`. Rdistance 'fitted objects' are typically produced by calls to `dfuncEstim`. 'Fitted objects' are 'model frames' with additional components such as the parameters estimates, log likelihood value, convergence information, and the variance- covariance matrix of the parameters.

Value

A list containing the following components

`start`	Vector of starting values for parameters of the likelihood and expansion terms.
`lowlimit`	Vector of lower limits for the likelihood parameters and expansion terms.
`uplimit`	Vector of upper limits for the likelihood parameters and expansion terms.
`names`	Vector of names for the likelihood parameters and expansion terms.

The length of each vector in the return is: `(Num expansions) + 1 + 1*(like %in% c("hazrate")) + (Num Covars)`.

See Also

dfuncEstim

Examples

```
data(sparrowDf)

# Half-normal start limits
modList <- parseModel(
     data = sparrowDf
   , formula = dist ~ 1
   , likelihood = "halfnorm"
)
startLimits(modList)

# Half-normal with expansions
modList <- parseModel(
     data = sparrowDf
   , formula = dist ~ 1
   , likelihood = "halfnorm"
   , expansions = 3
)
startLimits(modList)

# Hazard rate start limits
modList$likelihood <- "hazrate"
startLimits(modList)

# Neg exp start limits
modList$likelihood <- "negexp"
startLimits(modList)
```

---

summary.abund *Summarize abundance estimates*

Description

Summarize an object of class c("abund","dfunc") that is output by abundEstim.

Usage

```
## S3 method for class 'abund'
summary(object, criterion = "AICc", ...)
```

Arguments

object	An Rdistance model frame or fitted distance function, normally produced by a call to dfuncEstim.
criterion	A string specifying the model fit criterion to print. Must be one of "AICc" (the default), "AIC", or "BIC". See AIC.dfunc for formulas.
...	Included for compatibility to other print methods. Ignored here.

Details

If the proportion of bootstrap iterations that failed is greater than getOption("Rdistance_maxBSFailPropForWarning"), a warning about the validity of CI's is issued and a diagnostic message printed. Increasing this option to a number greater than 1 will kill the warning (e.g., options(Rdistance_maxBSFailPropForWarning = 1.3)), but ignoring a large number of non-convergent bootstrap iterations may be a bad idea (i.e., validity of the CI is questionable). The default value for Rdistance_maxBSFailPropForWarning is 0.2.

Value

0 is invisibly returned.

See Also

dfuncEstim, abundEstim, summary.dfunc, print.dfunc, print.abund

Examples

```
# Load example sparrow data (line transect survey type)
data(sparrowDf)

# Fit half-normal detection function
dfunc <- sparrowDf |> dfuncEstim(formula=dist ~ 1 + offset(groupsize))

# Estimate abundance given the detection function
fit <- abundEstim(dfunc
                , area = units::set_units(4105, "km^2")
                , ci=NULL)
summary(fit) # No confidence intervals

## Not run:
# With bootstrap confidence intervals
# Requires ~3 min to complete
fit <- abundEstim(dfunc
                , area = units::set_units(4105, "km^2")
                , ci=0.95)

summary(fit)

## End(Not run)
```

summary.dfunc *Summarize a distance function object*

Description

A summary method for distance functions. Distance functions are produced by dfuncEstim (class dfunc).

Usage

```
## S3 method for class 'dfunc'
summary(object, criterion = "AICc", ...)
```

Arguments

object	An Rdistance model frame or fitted distance function, normally produced by a call to dfuncEstim.
criterion	A string specifying the model fit criterion to print. Must be one of "AICc" (the default), "AIC", or "BIC". See AIC.dfunc for formulas.
...	Included for compatibility with other print methods. Ignored here.

Details

This function prints the following quantities:

- 'Call' : The original function call.
- 'Coefficients' : A matrix of estimated coefficients, their standard errors, and Wald Z tests.
- 'Strip' : The left (w.lo) and right (w.hi) truncation values.
- 'Effective strip width or detection radius' : ESW or EDR as computed by effectiveDistance.
- 'Probability of Detection' : Probability of detecting a single target in the strip.
- 'Scaling' : The horizontal and vertical coordinates used to scale the distance function. Usually, the horizontal coordinate is 0 and the vertical coordinate is 1 (i.e., g(0) = 1).
- 'Log likelihood' : Value of the maximized log likelihood.
- 'Criterion' : Value of the specified fit criterion (AIC, AICc, or BIC).

The number of digits used in the printout is controlled by options()$digits.

Value

The input distance function object (object), invisibly, with the following additional components:

- convMessage: The convergence message. If the distance function is smoothed, the convergence message is NULL.
- effDistance: The ESW or EDR.
- pDetect: Probability of detection in the strip.

- AIC: AICc, AIC, or BIC of the fit, whichever was requested.
- coefficients: If the distance function has coefficients, this is the coefficient matrix with standard errors, Wald Z values, and p values. If the distance function is smoothed, it has no coefficients and this component is NULL.

See Also

dfuncEstim, plot.dfunc, print.abund, print.abund

Examples

```
# Load example sparrow data (line transect survey type)
data(sparrowDf)

# Fit half-normal detection function
dfunc <- sparrowDf |> dfuncEstim(formula=dist~1)

# Print results
summary(dfunc)
summary(dfunc, criterion="BIC")
```

---

summary.rowwise_df *summary.rowwise_df - Summary method for Rdistance data frames*

Description

Summary method for distance sampling data frames. Rdistance data frames are rowwise tibbles. This routine is a replacement summary method for rowwise_df's that provides useful distance sampling descriptive statistics.

Usage

```
## S3 method for class 'rowwise_df'
summary(object, formula = NULL, w.lo = 0, w.hi = NULL, ...)
```

Arguments

object	An RdistDf data frame.
formula	A standard formula object. For example, dist ~ 1, dist ~ covar1 + covar2). The left-hand side (before ~) is the name of the vector containing off-transect or radial detection distances. The right-hand side contains the names of covariate vectors to fit in the detection function, and potentially group sizes. Covariates can be either detection level or transect level and can appear in data or exist in the global working environment. Regular R scoping rules apply.

w.lo	Lower or left-truncation limit of the distances in distance data. This is the minimum possible off-transect distance. Default is 0. If w.lo is greater than 0, it must be assigned measurement units using units(w.lo) <- "<units>" or w.lo <- units::set_units(w.lo, "<units>"). See examples in the help for set_units.
w.hi	Upper or right-truncation limit of the distances in dist. This is the maximum off-transect distance that could be observed. If unspecified (i.e., NULL), right-truncation is set to the maximum of the observed distances. If w.hi is specified, it must have associated measurement units. Assign measurement units using units(w.hi) <- "<units>" or w.hi <- units::set_units(w.hi, "<units>"). See examples in the help for set_units.
...	Other arguments for summary methods.

Value

If object is an RdistDf, a data frame containing summary statistics relevant to distance sampling is returned invisibly. If formula is not specified, the number of distance observations and target detections is not returned because the distances, group sizes, and covariates are not known. If object is not an Rdistance data frame, return is the result of the next summary method.

Examples

```
data(thrasherDf)
summary(thrasherDf)
summary(thrasherDf
        , formula = dist ~ groupsize(groupsize)
        , w.hi = units::set_units(100,"m")
        )
```

---

thrasherDetectionData *Sage Thrasher detection data*

Description

Point transect data collected in central Wyoming from 120 points surveyed for Sage Thrashers by the Wyoming Cooperative Fish & Wildlife Research Unit in 2013.

Format

A data.frame containing 193 rows and 3 columns. Each row represents a detected group of thrashers. Column descriptions:

1. siteID: Factor (120 levels), the site or point where the detection was made.
2. groupsize: Number, the number of individuals within the detected group.
3. dist: Number, the radial distance (m) from the transect to the detected group. This is the distance used in analysis.

Source

The Sage Thrasher data are a subset of the data collected by Jason Carlisle and various field technicians for his Ph.D. from the Department of Ecology, University of Wyoming, in 2017. This portion of Jason's work was funded by the Wyoming Game and Fish Department through agreements with the University of Wyoming's Cooperative Fish & Wildlife Research Unit (2012).

References

Carlisle, J.D. 2017. The effect of sage-grouse conservation on wildlife species of concern: implications for the umbrella species concept. Dissertation. University of Wyoming, Laramie, Wyoming, USA.

Carlisle, J. D., A. D. Chalfoun, K. T. Smith, and J. L. Beck. 2018. Nontarget effects on songbirds from habitat manipulation for Greater Sage-Grouse: Implications for the umbrella species concept. *The Condor: Ornithological Applications* 120:439–455. doi:10.1650/CONDOR17200.1

See Also

`thrasherSiteData`

---

thrasherDf *Sage Thrasher detection data frame in Rdistance >4.0.0 format*

Description

Point transect data collected in central Wyoming on 120 points surveyed for Sage Thrashers by the Wyoming Cooperative Fish & Wildlife Research Unit in 2013.

Format

A rowwise tibble containing 120 rows and 8 columns, one of which (i.e., 'detections') contains nested data frames of detections. Each row represents one transect of one point.

A data.frame containing 120 rows and 6 columns. Each row represents a surveyed site. Each surveyed site is considered one transect of one point. Column descriptions:

1. `siteID`: Factor (120 levels), the site or point surveyed.
2. `detections`: An embedded (nested) data frame containing detections made at that point. Columns in the embedded data frame contain:
   (a) `groupsize`: The number of individuals in the detected group.
   (b) `dist`: The radial distance (m) from the transect to the detected group.
3. `observer`: Factor (six levels), identity of the observer who surveyed the point.
4. `bare`: Number, the mean bare ground cover (%) within 100 m of each point.
5. `herb`: Number, the mean herbaceous cover (%) within 100 m of each point.
6. `shrub`: Number, the mean shrub cover (%) within 100 m of each point.
7. `height`: Number, the mean shrub height (cm) within 100 m of each point.
8. `npoints`: The number of point counts on the transect.

Source

The sage thrasher data are a subset of data collected by Jason Carlisle and various field technicians for his Ph.D. from the Department of Ecology, University of Wyoming, in 2017. This portion of Jason's work was funded by the Wyoming Game and Fish Department through agreements with the University of Wyoming's Cooperative Fish & Wildlife Research Unit (2012).

References

Carlisle, J.D. 2017. The effect of sage-grouse conservation on wildlife species of concern: implications for the umbrella species concept. Dissertation. University of Wyoming, Laramie, Wyoming, USA.

Carlisle, J. D., A. D. Chalfoun, K. T. Smith, and J. L. Beck. 2018. Nontarget effects on songbirds from habitat manipulation for Greater Sage-Grouse: Implications for the umbrella species concept. *The Condor: Ornithological Applications* 120:439–455. doi:10.1650/CONDOR17200.1

See Also

`thrasherSiteData`, `thrasherDetectionData`, `RdistDf`

Examples

```
data(thrasherDf)

is.RdistDf(thrasherDf)

summary(thrasherDf,
  formula = dist ~ groupsize(groupsize)
)
```

`thrasherSiteData` *thrasherSiteData - Sage Thrasher site data.*

Description

Point transect data collected in central Wyoming from 120 points surveyed for Sage Thrashers by the Wyoming Cooperative Fish & Wildlife Research Unit in 2013.

Format

A data.frame containing 120 rows and 6 columns. Each row represents a surveyed site (point). Column descriptions:

1. `siteID`: Factor (120 levels), the site or point surveyed.
2. `observer`: Factor (six levels), identity of the observer who surveyed the point.
3. `bare`: Number, the mean bare ground cover (%) within 100 m of each point.
4. `herb`: Number, the mean herbaceous cover (%) within 100 m of each point.
5. `shrub`: Number, the mean shrub cover (%) within 100 m of each point.
6. `height`: Number, the mean shrub height (cm) within 100 m of each point.

Source

The Sage Thrasher data are a subset of data collected by Jason Carlisle and field technicians for his Ph.D. from the Department of Ecology, University of Wyoming, in 2017. This portion of Jason's work was funded by the Wyoming Game and Fish Department through agreements with the University of Wyoming's Cooperative Fish & Wildlife Research Unit (2012).

References

Carlisle, J.D. 2017. The effect of sage-grouse conservation on wildlife species of concern: implications for the umbrella species concept. Dissertation. University of Wyoming, Laramie, Wyoming, USA.

Carlisle, J. D., A. D. Chalfoun, K. T. Smith, and J. L. Beck. 2018. Nontarget effects on songbirds from habitat manipulation for Greater Sage-Grouse: Implications for the umbrella species concept. *The Condor: Ornithological Applications* 120:439–455. doi:10.1650/CONDOR17200.1

See Also

`thrasherDetectionData`

`transectType` *transectType - Type of transects*

Description

Return the type of transects represented in either a fitted distance function or Rdistance data frame.

Usage

```
transectType(x)
```

Arguments

`x`	Either an estimated distance function, output by `dfuncEstim`, or an Rdistance nested data frame, output by `RdistDf`.

Details

This function is a simple helper function. If `x` is an estimated distance object, it polls the `transType` attribute of `x`'s Rdistance nested data frame. If `x` is an Rdistance nested data frame, it polls the `transType` attribute.

Value

A scalar. Either 'line' if `x` contains continuous line-transect detections, or 'point' if `x` contains point-transects detections. If transect type has not been assigned, return is NULL.

unnest *unnest - Unnest an RdistDf data frame*

Description

Unnest an RdistDf data frame by expanding the embedded 'detections' column. This unnest includes the so-called zero transects (transects without detections).

Usage

```
unnest(data, ...)
```

Arguments

data	An RdistDf data frame. RdistDf data frames contain one line per transect and a list-based column. The list-based column contains a data frame with detection information. The detection information data frame on each row contains (at least) distances and group sizes of all targets detected on the transect. Function RdistDf creates RdistDf data frames from separate transect and detection data frames. is.RdistDf checks whether data frames are RdistDf's.
...	Additional arguments passed to tidyr::unnest if data is not an RdistDf.

Value

An expanded data frame, without embedded data frames. Rows in the return represent with one detection or one transect. If multiple detections were made on one transect, the transect will appear on multiple rows. If no detections were made on a transect, it will appear on one row with NA detection distance.

Examples

```
data('sparrowDf')

# tidyr::unnest() does not include zero transects
detectionDf <- tidyr::unnest(sparrowDf, detections)
nrow(detectionDf)
any(detectionDf$siteID == "B2")

# Rdistance::unnest() includes zero transects
fullDf <- unnest(sparrowDf)
nrow(fullDf)
any(fullDf$siteID == "B2")
```

Index