Package 'Rdistance'

June 13, 2023

Type Package

Title Distance-Sampling Analyses for Density and Abundance Estimation

Version 3.0.0

Date 2023-05-25

Maintainer Trent McDonald <trent@mcdonalddatasciences.com>

Description Distance-sampling analyses estimate density and abundance of organisms in ecology when detection probability declines with increasing distance from the observers. Distance-sampling is popular in most branches of ecology and especially when organisms are observed from aerial platforms (e.g., airplane or drone), surface vessels (e.g., boat or truck), or along walking transects. Rdistance analyzes data collected on both point and line transects, estimates overall (study area) and site-level (transect or point) density, and allows users to specify regression-like formula (similar to lm or glm) for covariates. A large suite of classical, parametric detection functions are included along with some uncommon parametric functions (e.g., Gamma, negative exponential) and non-parametric smoothed distance functions. Custom (user-defined) detection functions can be implemented (see vignette). Measurement unit integrity is enforced with internal unit conversion when necessary. The help files and vignettes have been vetted by multiple authors and tested in workshop settings.

License GNU General Public License

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

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

Suggests knitr, rmarkdown, testthat (>= 3.0.0), devtools

VignetteBuilder knitr

Depends R (>= 3.5.0)

Imports graphics, stats, utils, units, crayon

RoxygenNote 7.2.3

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 2023-06-13 07:30:05 UTC

R topics documented:

`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 covariate in distance functions (`dfuncEstim`).
- Automatic bootstrap confidence intervals (`abundEstim`).
- Availability of both study-area and site-level abundance estimates (`abundEstim`).
- Classical, parametric distance functions (`halfnorm.like`, `hazrate.like`), and expansion functions (`cosine.expansion`, `hermite.expansion`, `simple.expansion`).
- Non-classic distance functions (`Gamma.like`, `negexp.like`, `uniform.like`) and a non-parametric smoother `dfuncSmu`).
- User defined distance functions.

- Automated distance function fits and selection autoDistSamp.
- Extended vignettes.
- 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. The actual locations of sighted targets are often recorded or computed. 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 (sometimes called a point count) methods (i.e., thrasherDetectionData and thrasherSiteData).

These datasets demonstrate the type and format of input data required by Rdistance to estimate a detection function and abundance from distance sampling data collected by surveying line transects or point transects. They also allow the user to step through the tutorials described in the package vignettes.

Rdistance requires only detection data to fit detection functions, assuming no covariates in the detection function (see dfuncEstim). Both detection and site data are required to estimate abundance or to include site-level covariates in the detection function (see abundEstim).

Resources

The best place to start learning about Rdistance is at the package's GitHub Wiki, which hosts several tutorial vignettes and FAQs (https://github.com/tmcd82070/Rdistance/wiki). Additionally, the examples in the help files for dfuncEstim, abundEstim, and autoDistSamp highlight the package's primary functionality.

A list of routines can be obtained by loading Rdistance and issuing help(package="Rdistance").

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

---

abundEstim *Estimate abundance from distance-sampling data*

Description

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

Usage

```
abundEstim(
  dfunc,
  detectionData,
  siteData,
  area = NULL,
  singleSided = FALSE,
  ci = 0.95,
  R = 500,
  lengthColumn = "length",
  plot.bs = FALSE,
  showProgress = TRUE,
  control = RdistanceControls()
)
```

Arguments

dfunc	An estimated 'dfunc' object produced by dfuncEstim.
detectionData	A data frame containing detection distances (either perpendicular for line-transect or radial for point-transect designs), with one row per detected object or group. This data frame must contain at least the following information: • Detection Distances: A single column containing detection distances must be specified on the left-hand side of formula. As of Rdistance version 3.0.0, the detection distances must have measurement units attached. Attach measurements units to distances using library(units);units()<-. For example, library(units) followed by units(df$dist) <- "m" or units(df$dist) <- "ft" will work. Alternatively, df$dist <- units::set_units(df$dist, "m") also works. • Site IDs: The ID of the transect or point (i.e., the 'site') where each object or group was detected. The site ID column(s) (see arguments transectID and pointID) must specify the site (transect or point) so that this data frame can be merged with siteData. • In a later release, Rdistance will allow detection-level covariates. When that happens, detection-level covariates will appear in this data frame. See example data set sparrowDetectionData. See also **Input data frames** below for information on when detectionData and siteData are required inputs.
siteData	A data.frame containing site (transect or point) IDs and any *site level* covariates to include in the detection function. Every unique surveyed site (transect or point) is represented on one row of this data set, whether or not targets were sighted at the site. See arguments transectID and pointID for an explanation of the way in which distance and site data frames are merged. See section **Relationship between data frames (transect and point ID's)** for additional details. See **Data frame requirements** for situations in which detectionData only, detectionData and siteData, or neither are required.
area	A scalar containing the total area of inference. Commonly, this is study area size. If area is NULL (the default), area will be set to 1 square unit of the output units and this produces abundance estimates equal density estimates. 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 many others are acceptable.
singleSided	Logical scaler. If only one side of the transect was observed, set singleSided = TRUE. If both sides of line-transects were observed, singleSided = FALSE. Some surveys observe only one side of transect lines for a variety of logistical reasons. For example, some aerial line-transect surveys place observers on only one side of the aircraft. This parameter effects only line-transects. When singleSided = TRUE, surveyed area is halved and the density estimator's denominator (see **Details**) is $(ESW)(L)$, not $2(ESW)(L)$.

ci	A scalar indicating the confidence level of confidence intervals. Confidence intervals are computed using a bias corrected bootstrap method. If ci = NULL, confidence intervals are not computed.
R	The number of bootstrap iterations to conduct when ci is not NULL.
lengthColumn	Character string specifying the (single) column in siteData that contains transect lengths. This is ignored if pointSurvey = TRUE. This column must have measurement units.
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.
control	A list containing optimization control parameters such as the maximum number of iterations, tolerance, the optimizer to use, etc. See the RdistanceControls function for explanation of each value, the defaults, and the requirements for this list. See examples below for how to change controls.

Details

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

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

where *n* is total number of sighted individuals (i.e., sum(dfunc$detections$groupSizes)), *L* is the total length of surveyed transect (i.e., sum(siteData[,lengthColumn])), 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, *P* is the total number of surveyed points, 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.

Value

An '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:

density	Estimated density on the sampled area with units. The *effectively* sampled area is 2*L*ESW (not 2*L*w.hi). Density has squared units of the requested output units. Convert density to other units with units::set_units(x$density, "<units>").

n.hat	Estimated abundance on the study area (if area > 1) or estimated density on the study area (if area = 1), without units.
n	The number of detections (not individuals, unless all group sizes = 1) on non-NA length transects used to compute density and abundance.
n.seen	The total number of individuals seen on transects with non-NA length. Sum of group sizes used to estimate density and abundance.
area	Total area of inference in squared output units.
surveyedUnits	The total length of sampled transect with units. This is the sum of the lengthColumn column of siteData.
avg.group.size	Average group size on transects with non-NA length transects.
rng.group.size	Minimum and maximum groupsizes observed on non-NA length transects.
effDistance	A vector containing effective sample distance. If covariates are not included, length of this vector is 1 because effective sampling distance is constant over detections. If covariates are included, this vector has length equal to the number of detections (i.e., x$n). This vector was produced by a call to effectiveDistance() with newdata set to NULL.
n.hat.ci	A vector containing the lower and upper limits of the bias corrected bootstrap confidence interval for abundance.
density.ci	A vector containing the lower and upper limits of the bias corrected bootstrap confidence interval for density, with units.
effDistance.ci	A vector containing the lower and upper limits of the bias corrected bootstrap confidence interval for *average* effective sampling distance.
B	A data frame containing bootstrap values of coefficients, density, and effective distances. Number of rows is always R, the requested number of bootstrap iterations. If a particular iteration did not converge, 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. The second to last column contains bootstrap values of density (with units). The last column of B contains bootstrap values of effective sampling distance or radius (with units). If the distance function contains covariates, the effective sampling distance column is the average effective distance over detections used during the associated bootstrap iteration.
nItersConverged	The number of bootstrap iterations that converged.
alpha	The (scalar) confidence level of the confidence interval for n.hat.

Bootstrap Confidence Intervals

The bootstrap confidence interval for abundance assumes that the fundamental units of replication (lines or points, hereafter "sites") are independent. The bias corrected bootstrap method used here resamples the units of replication (sites), refits the distance function, and estimates abundance using the resampled counts and re-estimated distance function. The original data frames, detectionData and siteData, are needed here for bootstrapping because they contain the transect and detection information. If a double-observer data frame is included in dfunc, rows of the double-observer data frame are re-sampled each bootstrap iteration.

This routine does not re-select the distance model fitted to resampled data. The model in the input object is re-fitted every iteration.

By default, R = 500 iterations are performed, after which the bias corrected confidence intervals are computed (Manly, 1997, section 3.4).

During bootstrap iterations, the distance function can fail to converge on the resampled data. An iteration can fail to converge for a two reasons: (1) no detections on the iteration, and (2) bad configuration of distances on the iteration which pushes parameters to their bounds. When an iteration fails to produce a valid distance function, Rdistance simply skips the intration, effectively ignoring these non-convergent iterations. If the proportion of non-convergent iterations is small (less than 20 on abundance is probably valid. If the proportion of non-convergent iterations is not small (exceeds 20 The print method (print.abund) is the routine that issues this warning. The warning can be turned off by setting maxBSFailPropForWarning in the print method to 1.0, or by modifying the code in RdistanceControls() to re-set the default threshold and storing the modified function in your .GlobalEnv. Additional iterations may be needed to achieve an adequate number. Check number of convergent iterations by counting non-NA rows in output data frame 'B'.

Missing Transect Lengths

Line transects: The transect length column of siteData can contain missing values. NA length transects are equivalent to 0 [m] transects and do not count toward total surveyed units. NA length transects are handy if some off-transect distance observations should be included when estimating the distance function, but not when estimating abundance. To do this, include 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. Group sizes associated with NA length transects are dropped and not counted toward density or abundance. Among other things, this allows estimation of abundance on one study area using off-transect distance observations from another.

Point transects: Point transects do not have length. The "length" of point transects is the number of points on the transect. Rdistance treats individual points as independent and bootstrap resampmles them to estimate variance. To include distance obervations from some points but not the number of targets seen, include a separate "length" column in the site data frame with NA for the "extra" points. Like NA length line transects, NA "length" point transects are dropped from the count of points and group sizes on these transects are dropped from the counts of targets. This allows users to estimate their distance function on one set of observations while inflating counts from another set of observations. A transect "length" column is not required for point transects. Values in the lengthColumn do not matter except for NA (e.g., a column of 1's mixed with NA's is acceptable).

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.

Examples

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

# Fit half-normal detection function
dfunc <- dfuncEstim(formula=dist ~ groupsize(groupsize)
                    , detectionData=sparrowDetectionData
                    , likelihood="halfnorm"
                    , w.hi=units::set_units(100, "m")
                    )

# Estimate abundance given a detection function
# No variance on density or abundance estimated here
# due to time constraints.  Set ci=0.95 (or another value)
# to estimate bootstrap variances on ESW, density, and abundance.

fit <- abundEstim(dfunc
                , detectionData = sparrowDetectionData
                , siteData = sparrowSiteData
                , area = units::set_units(4105, "km^2")
                , ci = NULL
                )
```

`AIC.dfunc` *AICc and related fit statistics for detection function objects*

Description

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

Usage

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

Arguments

`object`	An estimated detection function object. An estimated detection function object has class 'dfunc', and is usually produced by a call to `dfuncEstim`.
`...`	Required for compatibility with the general `AIC` method. Any extra arguments to this function are ignored.
`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. By default, the value of AICc for the estimated distance function `obj`.

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(sparrowDetectionData)
dfunc <- dfuncEstim(dist~1,
                    detectionData=sparrowDetectionData,
                    w.hi=units::set_units(150, "m"))

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

`autoDistSamp` *Automated classical distance analysis*

Description

Perform automated classical detection function selection and estimation of abundance.

Usage

```
autoDistSamp(
  formula,
  detectionData,
  siteData,
  w.lo = 0,
  w.hi = NULL,
  likelihoods = c("halfnorm", "hazrate", "uniform", "negexp", "Gamma"),
  series = c("cosine", "hermite", "simple"),
  expansions = 0:3,
  pointSurvey = FALSE,
  warn = TRUE,
  area = NULL,
  ci = 0.95,
  R = 500,
  plot.bs = FALSE,
  showProgress = TRUE,
  plot = TRUE,
  criterion = "AICc",
  ...
)
```

Arguments

`formula`	This parameter is passed to `dfuncEstim`. See `dfuncEstim` documentation for definition.
`detectionData`	This parameter is passed to `dfuncEstim` and `abundEstim`. See `abundEstim` documentation for definition.
`siteData`	This parameter is passed to `abundEstim`. See `abundEstim` documentation for definition.
`w.lo`	This parameter is passed to `dfuncEstim`. See `dfuncEstim` documentation for definition.
`w.hi`	This parameter is passed to `dfuncEstim`. See `dfuncEstim` documentation for definition.
`likelihoods`	Vector of strings specifying the likelihoods to consider during model selection. Valid values at present are "uniform", "halfnorm", "hazrate", "negexp", and "Gamma". See Details for the models this routine considers.

series	Vector of series types to consider during model selection. Valid values are 'simple', 'hermite', and 'cosine'. See Details for the models this routine considers.
expansions	Vector of the number of expansion terms to consider during model selection. Valid values are 0 through 3. See Details for the models this routine considers. Note, expansion terms are not currently allowed in models with covariates.
pointSurvey	This parameter is passed to dfuncEstim. See dfuncEstim documentation for definition.
warn	This parameter is passed to dfuncEstim. dfuncEstim documentation for definition.
area	This parameter is passed to abundEstim. See abundEstim documentation for definition.
ci	This parameter is passed to abundEstim. See abundEstim documentation for definition.
R	This parameter is passed to abundEstim. See abundEstim documentation for definition.
plot.bs	Logical for whether to plot bootstrap iterations after the top model has been selected and during final estimation of confidence intervals. This parameter is passed unchanged to abundEstim. See abundEstim help for additional information.
showProgress	Logical for whether to suppress intermediate output. If showProgress=TRUE, a table of model fitting results appears in the console as they are estimated, and a progress bar shows progress through the bootstrap iterations at the end. If showProgress=FALSE, all intermediate output is suppressed which is handy for programming and simulations.
plot	Logical scalar specifying whether to plot models during model selection. If TRUE, a histogram with fitted distance function is plotted for every fitted model. The function pauses between each plot and prompts the user for whether they want to continue or not. For completely automated estimation, set plot = FALSE.
criterion	A string specifying the criterion to use when assessing model fit. The best fitting model from this routine is the one with lowest value of this fit criterion. This must be one of "AICc" (the default), "AIC", or "BIC". See AIC.dfunc for formulas.
...	Additional parameters passed to dfuncEstim, which in turn are passed to F.gx.estim. These include x.scl, g.x.scl, and observer for estimating double observer probabilities.

Details

During model selection, each series and number of expansions is crossed with each of the likelihoods. For example, if likelihoods has 3 elements, series has 2 elements, and expansions has 4 elements, the total number of models fitted is 3 (likelihoods) * 2 (series) * 4 (expansions) = 24 models. The default specification fits 41 detection functions from the "halfnorm", "hazrate", "uniform", "negexp", and "Gamma" likelihoods (note that Gamma does not currently implement expansions, see Gamma.like). Note, expansion terms are not currently allowed in models with covariates. The model with lowest AIC is selected as 'best', and estimation of abundance proceeds using that model.

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

Value

If bySite==FALSE, an 'abundance estimate' object is returned. See abundEstim and dfuncEstim for an explanation of components. Returned abundance estimates are based on the best fitting distance function among those fitted. A fit table, sorted by the criterion, is returned as component $fitTable. The fit table component contains columns like (likelihood), series, expansions, converge (0=converged,1=not), scale (1=passed scale check,0=did not pass), and aic (the criterion used).

If bySite==TRUE, a data frame containing site-level abundance based on the best-fitting detection function is returned. See abundEstim for description of columns in the data frame. The best-fitting likelihood form, series, and number of expansions are returned as attributes of the data frame (e.g., best-fitting likelihood is attr(out,"like.form")).

See Also

dfuncEstim, abundEstim.

Examples

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

# Automate fitting multiple detection functions, and estimate abundance
# (density per ha in this case), given the 'best' detection function
autoDistSamp(formula = dist ~ groupsize(groupsize)
           , detectionData = sparrowDetectionData
           , siteData = sparrowSiteData
           , likelihood = c("halfnorm", "hazrate")
           , w.hi = units::set_units(100, "m")
           , expansions = 0
           , area = units::set_units( 4105, "km^2" )
           , ci = NULL
           , plot = FALSE
           )
```

---

coef.dfunc *Coefficients of an estimated detection function*

Description

Extract the coefficients and estimated parameters (if any) from a estimated detection function object.

Usage

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

Arguments

object	An estimated distance function object. An estimated distance function object has class 'dfunc', and is usually produced by a call to dfuncEstim.
...	Required for compatibility with the general coef method. Any extra arguments to this function are ignored.

Details

This is an extractor function for the parameters of an estimated detection function. This function is equivalent to obj$parameters for classical detection functions.

Value

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

See Also

AIC, dfuncEstim

Examples

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

# Fit half-normal detection function
dfunc <- dfuncEstim(formula=dist~1
                  , detectionData=sparrowDetectionData)

# Print results
dfunc

# Extract the coefficient(s)
coef(dfunc)
```

---

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 *calculation of cosine expansion for detection function likelihoods*

Description

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

Usage

```
cosine.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, 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

```
set.seed(33328)
  x <- rnorm(1000) * 100
  x <- x[ 0 < x & x < 100 ]
  cos.expn <- cosine.expansion(x, 5)
```

dfuncEstim *Estimate a detection function from distance-sampling data*

Description

Fit a specific detection function to off-transect or off-point (radial) distances using maximum likelihood. Distance functions are fitted to individual distance observations, not histogram bin heights, despite plot methods that draw histogram bars.

Usage

```
dfuncEstim(
  formula,
  detectionData,
  siteData,
  likelihood = "halfnorm",
  pointSurvey = FALSE,
  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,
  observer = "both",
  warn = TRUE,
  transectID = NULL,
  pointID = "point",
  outputUnits = NULL,
  control = RdistanceControls()
)
```

Arguments

formula	A standard formula object (e.g., `dist ~ 1`, `dist ~ covar1 + covar2`). The left-hand side (before ~) is the name of the vector containing distances (off-transect or radial). The right-hand side (after ~) contains the names of covariate vectors to fit in the detection function. Covariates can be either detection level and appear in `detectionData` or transect level and appear in `siteData`. Regular R scoping rules apply. **Group Sizes:** Non-unity group sizes are specified using `groupsize()` in the formula. That is, when group sizes are not all 1, they must be entered as a column in `detectionData` and specified using `groupsize()` as part of `formula`. For example, `d ~ habitat + groupsize(number)` specifies that distances appear in variable `d`, one covariate named `habitat` is to be fitted, and 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.

`detectionData` A data frame containing detection distances (either perpendicular for line-transect or radial for point-transect designs), with one row per detected object or group. This data frame must contain at least the following information:

- Detection Distances: A single column containing detection distances must be specified on the left-hand side of `formula`. As of Rdistance version 3.0.0, the detection distances must have measurement units attached. Attach measurements units to distances using `library(units);units()<-`. For example, `library(units)` followed by `units(df$dist) <- "m"` or `units(df$dist) <- "ft"` will work. Alternatively, `df$dist <- units::set_units(df$dist, "m")` also works.
- Site IDs: The ID of the transect or point (i.e., the 'site') where each object or group was detected. The site ID column(s) (see arguments `transectID` and `pointID`) must specify the site (transect or point) so that this data frame can be merged with `siteData`.
- In a later release, `Rdistance` will allow detection-level covariates. When that happens, detection-level covariates will appear in this data frame.

See example data set `sparrowDetectionData`. See also **Input data frames** below for information on when `detectionData` and `siteData` are required inputs.

`siteData` A data.frame containing site (transect or point) IDs and any *site level* covariates to include in the detection function. Every unique surveyed site (transect or point) is represented on one row of this data set, whether or not targets were sighted at the site. See arguments `transectID` and `pointID` for an explanation of the way in which distance and site data frames are merged. See section **Relationship between data frames (transect and point ID's)** for additional details.

See **Data frame requirements** for situations in which `detectionData` only, `detectionData` and `siteData`, or neither are required.

`likelihood` String specifying the likelihood to fit. Built-in likelihoods at present are "uniform", "halfnorm", "hazrate", "negexp", and "Gamma". See vignette for a way to use user-define likelihoods.

`pointSurvey` A logical scalar specifying whether input data come from point-transect surveys (TRUE), or line-transect surveys (FALSE).

`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 to scale the sightability function to `g.x.scl`, 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. See Details for more on scaling the sightability function.

`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. See Details.

`observer` A numeric scalar or text string specifying whether observer 1 or observer 2 or both were full-time observers. This parameter dictates which set of observations form the denominator of a double observer system. If, for example, observer 2 was a data recorder and part-time observer, or if observer 2 was the pilot, set `observer` = 1. If `observer` = 1, observations by observer 1 not seen by observer 2 are ignored. The estimate of detection in this case is the ratio of number of targets seen by both observers to the number seen by both plus the number seen by just observer 2. If observer = "both", the computation goes both directions.

`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`.

`transectID` A character vector naming the transect ID column(s) in `detectionData` and `siteData`. If transects are not identified in columns named 'siteID' (the default for both data frames), you need to specify which column(s) uniquely identify transects. `transectID` can have length greater than 1, in which case unique transects are identified by the composite columns.

`pointID` When point-transects are used, this is the ID of points on a transect. When `pointSurvey=TRUE`, the combination of `transectID` and `pointID` specify unique sampling sites. See **Input data frames**.

If single points are surveyed, meaning surveyed points were not grouped into transects, each 'transect' consists of one point. In this case, set `transectID` equal to the point's ID and set `pointID` equal to 1 for all points.

`outputUnits` A string giving the symbolic measurment units that results should be reported in. Any distance measurement unit in `units::valid_udunits()` will work. The strings for common distance symbolic units are: "m" for meters, "ft" for feet, "cm" for centimeters, "mm" for millimeters, "mi" for miles, "nmile" for nautical miles ("nm" is nano meters), "in" for inches, "yd" for yards, "km" for kilometers, "fathom" for fathoms, "chains" for chains, and "furlong" for furlongs. If `outputUnits` is unspecified (NULL), output units are the same as distance measurements units in `data`.

`control` A list containing optimization control parameters such as the maximum number of iterations, tolerance, the optimizer to use, etc. See the `RdistanceControls` function for explanation of each value, the defaults, and the requirements for this list. See examples below for how to change controls.

Value

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

`parameters` 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 either 'hazrate' or 'uniform' (hazrate and uniform have two parameters).

`varcovar` The variance-covariance matrix for coefficients of the distance function, estimated by the inverse of the Hessian of the fit evaluated at the estimates. There is no guarantee this matrix is positive-definite and should be viewed with caution. Error estimates derived from bootstrapping are generally more reliable.

`loglik` The maximized value of the log likelihood (more specifically, the minimized value of the negative log likelihood).

`convergence` The convergence code. This code is returned by `optim`. Values other than 0 indicate suspect convergence.

`like.form` 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.

`detections` A data frame of detections within the strip or circle used in the fit. Column 'dist' contains the observed distances. Column 'groupSize' contains group sizes associated with the values of 'dist'. Group sizes are only used in `abundEstim`. This data frame contains only distances between `w.lo` and `w.hi`. Another component of the returned object, i.e., `model.frame` contains all observations in the input data, including those outside the strip.

`covars` Either NULL if no covariates are included in the detection function, or a `model.matrix` containing the covariates used in the fit. Factors in in the model.matrix version have been expanded into 0-1 indicator variables based on R contrasts in effect at the time of the call. Only covariates associated with distances inside the strip or circle are included.

`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.

Transect types

Rdistance accommodates two kinds of transects: continuous and point. On continuous transects detections can occur at any point along the route, and these are line-transects. On point transects detections can only occur at a series of stops (points), and these are point-transects. Transects are the basic sampling unit in both cases. Columns named in transectID are sufficient to specify unique line-transects. The combination of transectID and pointID specify unique sampling locations along point-transects. See **Input data frames** below for more detail.

Input data frames

To save space and to easily specify sites without detections, all site ID's, regardless of whether a detection occurred there, and *site level* covariates are stored in the siteData data frame. Detection distances and group sizes are measured at the *detection level* and are stored in the detectionData data frame.

Data frame requirements: The following explains conditions under which various combinations of the input data frames are required.

1. **Detection data and site data both required:**
   Both detectionData and siteData are required if *site level* covariates are specified on the right-hand side of formula. *Detection level* covariates are not currently allowed. Both detectionData and siteData data frames are required to estimate abundance later in abundEstim.

2. **Detection data only required:**
   detectionData only is required when covariates are are not included in the distance function (i.e., the right-hand side of formula is "~1" or "~groupsize(groupSize)"). Note that dfuncEstim does not need to know transect IDs (or group sizes) in order to estimate a distance function; but, group sizes and transect IDs are stored and used to estimate abundance in function abundEstim. Both the detectionData and siteData data frames are required in abundEstim.
3. **Neither detection data nor site data required**
   Neither detectionData nor siteData are required if all variables specified in formula are within the scope of dfuncEstim (e.g., in the global working environment) and abundance estimates are not required. Regular R scoping rules apply when the call to dfuncEstim is embedded in a function. This case is will produce distance functions only. Abundance cannot later be estimated because transects and transect lengths cannot be specified outside of a data frame. If abundance will be estimated, use either case 1 or 2.

Relationship between data frames (transect and point ID's): The input data frames, detectionData and siteData, must be merge-able on unique sites. For line-transects, site ID's specify transects or routes and are unique values of the transectID column in siteData. In this case, the following merge must work: merge(detectionData,siteData,by=transectID).

For point-transects, site ID's specify individual points and are unique values of the combination paste(transectID,pointID). In this case, the following merge must work: merge(detectionData,siteData,by=c(tra
pointID).

By default, *transects* are unique combinations of the common variables in the detectionData and siteData data frames if both data frames are specified (i.e., unique values of intersect(names(detectionData), names(siteData))). If siteData is not specified and transectID is not given, transects are assumed to be identified in a column named siteID in detectionData.

Either way (i.e., either transectID = "siteID" or specified as something else), the column(s) containing transect ID's must be correct here if abundance is to be estimated later. Routine abundEstim requires transect ID's for bootstrapping because it resamples unique values of the composite transect ID column(s). abundEstim uses the value of transectID specified here and hence users cannot change transect ID's between calls to dfuncEstim and abundEstim and all transectID columns must be present in both data frames even though they may not be used until later.

An error occurs if both detectionData and siteData are specified but no common columns exist. Duplicate transectID values are not allowed in siteData but are allowed in detectionData because multiple detections can occur on a single transect or at a single site. If the same site is surveyed in multiple years, specify another level of transect ID; for example, transectID = c("year","transectID").

Measurement Units

As of Rdistance version 3.0.0, measurement units are require on all distances. This includes off-transect distances, radial distances, truncation distances (w.lo and w.hi), transect lengths, and study size area. In dfuncEstim, units are required on the following: detectionData$dist; w.lo (unless it is zero); w.hi (unless it is NULL); and x.scl. In abundEstim, units are required on siteData$length and area. All units are 1-dimensional except those on area, which are 2-dimensional.

Requiring units ensures that internal calculations and results (e.g., ESW and abundance) are correct and that output units are clear. Input distances can have variable units. For example, input distances can be in specified in "m", `w.hi` in "in", and `w.lo` in "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`.

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 `RdistanceControls(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`). See package vignettes for additional options.

Examples

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

dfunc <- dfuncEstim(formula = dist ~ 1
                  , detectionData = sparrowDetectionData)
dfunc
plot(dfunc)
```

`dfuncSmu` *Estimate a non-parametric smooth detection function from distance-sampling data*

Description

Estimates a smooth detection function for line-transect perpendicular distances or point-transect radial distances.

Usage

```
dfuncSmu(
  formula,
  detectionData,
  siteData,
  bw = "SJ-dpi",
  adjust = 1,
  kernel = "gaussian",
  pointSurvey = FALSE,
  w.lo = units::set_units(0, "m"),
  w.hi = NULL,
  x.scl = "max",
  g.x.scl = 1,
  observer = "both",
  warn = TRUE,
  transectID = NULL,
  pointID = "point",
  outputUnits = NULL,
  length = "length",
  control = RdistanceControls()
)
```

Arguments

`formula` A formula object (e.g., dist ~ 1). The left-hand side (before ~) is the name of the vector containing distances (perpendicular or radial). The right-hand side (after ~) must be the intercept-only model as `Rdistance` does not currently allow covariates in smoothed distance functions. If names in `formula` do not appear in `detectionData`, the normal scoping rules for model fitting routines (e.g., `lm` and `glm`) apply.

`detectionData` A data frame containing detection distances (either perpendicular for line-transect or radial for point-transect designs), with one row per detected object or group. This data frame must contain at least the following information:

- Detection Distances: A single column containing detection distances must be specified on the left-hand side of `formula`.
- Site IDs: The ID of the transect or point (i.e., the 'site') where each object or group was detected. The site ID column(s) (see argument `siteID`) must specify the site (transect or point) so that this data frame can be merged with `siteData`.

Optionally, this data frame can contain the following variables:

- Group Sizes: The number of individuals in the group associated with each detection. If unspecified, `Rdistance` assumes all detections are of single individuals (i.e., all group sizes are 1).
- When `Rdistance` allows detection-level covariates in some version after 2.1.1, detection-level covariates will appear in this data frame.

See example data set sparrowDetectionData). See also **Input data frames** below for information on when detectionData and siteData are required inputs.

siteData — A data.frame containing site (transect or point) IDs and any *site level* covariates to include in the detection function. Every unique surveyed site (transect or point) is represented on one row of this data set, whether or not targets were sighted at the site. See arguments transectID and pointID for an explanation of site and transect ID's.

If sites are transects, this data frame must also contain transect length. By default, transect length is assumed to be in column 'length' but can be specified using argument length.

The total number of sites surveyed is nrow(siteData). Duplicate site-level IDs are not allowed in siteData.

See **Input data frames** for when detectionData and siteData are required inputs.

bw — Bandwidth of the smooth, which controls smoothness. Smoothing is done by stats::density, and bw is passed straight to it's bw argument. bw can be numeric, in which case it is the standard deviation of the Gaussian smoothing kernel. Or, bw can be a character string specifying the bandwidth selection rule. Valid character string values of bw are the following:

- "nrd0" : Silverman's 'rule-of-thumb' equal to $\frac{0.9s}{1.34n^{-0.2}}$, where s is the minimum of standard deviation of the distances and the interquartile range. See bw.nrd0.
- "nrd" : The more common 'rule-of-thumb' variation given by Scott (1992). This rule uses 1.06 in the denominator of the "nrd0" bandwidth. See bw.nrd
- "bcv" : The biased cross-validation method. See bcv.
- "ucv" : The unbiased cross-validation method. See ucv.
- "SJ" or "SJ-ste" : The 'solve-the-equation' bandwidth of Sheather & Jones (1991). See bw.SJ or width.SJ.
- "SJ-dpi" (default) : The 'direct-plug-in' bandwidth of Sheather & Jones (1991). See bw.SJ or width.SJ.

adjust — Bandwidth adjustment for the amount of smooth. Smoothing is done by density, and this parameter is passed straight to it's adjust argument. In stats::density, the bandwidth used is actually adjust*bw, and inclusion of this parameters makes it easier to specify values like 'half the default' bandwidth.

kernel — Character string specifying the smoothing kernel function. This parameters is passed unmodified to stats::density. Valid values are:

- "gaussian" : Gaussian (normal) kernel, the default
- "rectangular" : Uniform or flat kernel
- "triangular" : Equilateral triangular kernel
- "epanechnikov" : the Epanechnikov kernel
- "biweight" : the biweight kernel
- "cosine" : the S version of the cosine kernel
- "optcosine" : the optimal cosine kernel which is the usual one reported in the literature

Values of kernel may be abbreviated to the first letter of each string. The numeric value of bw used in the smooth is stored in the $fit component of the returned object (i.e., in returnedfitbw).

pointSurvey — A logical scalar specifying whether input data come from point-transect surveys (TRUE), or line-transect surveys (FALSE). Point surveys (TRUE) have not been implemented yet.

w.lo — Lower or left-truncation limit of the distances in distance data. This is the minimum possible off-transect distance. Default is 0.

w.hi — Upper or right-truncation limit of the distances in dist. This is the maximum off-transect distance that could be observed. If left unspecified (i.e., at the default of NULL), right-truncation is set to the maximum of the observed distances.

x.scl — This parameter is passed to F.gx.estim. See F.gx.estim documentation for definition.

g.x.scl — This parameter is passed to F.gx.estim. See F.gx.estim documentation for definition.

observer — This parameter is passed to F.gx.estim. See F.gx.estim documentation for definition.

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, messages about convergence and boundary conditions are printed by print.dfunc, print.abund, and plot.dfunc, so there should be little harm in setting warn = FALSE.

transectID — A character vector naming the transect ID column(s) in detectionData and siteData. Transects can be the basic sampling unit (when pointSurvey=FALSE) or contain multiple sampling units (e.g., when pointSurvey=TRUE). For line-transects, the transectID column(s) alone is sufficient to specify unique sample sites. For point-transects, the amalgamation of transectID and pointID specify unique sampling sites. See **Input data frames**.

pointID — When point-transects are used, this is the ID of points on a transect. When pointSurvey=TRUE, the amalgamation of transectID and pointID specify unique sampling sites. See **Input data frames**.

If single points are surveyed, meaning surveyed points were not grouped into transects, each 'transect' consists of one point. In this case, set transectID equal to the point's ID and set pointID equal to 1 for all points.

outputUnits — A string giving the symbolic measurment units that results should be reported in. Any distance measurement unit in units::valid_udunits() will work. The strings for common distance symbolic units are: "m" for meters, "ft" for feet, "cm" for centimeters, "mm" for millimeters, "mi" for miles, "nmile" for nautical miles ("nm" is nano meters), "in" for inches, "yd" for yards, "km" for kilometers, "fathom" for fathoms, "chains" for chains, and "furlong" for furlongs. If outputUnits is unspecified (NULL), output units are the same as distance measurements units in data.

length	Character string specifying the (single) column in siteData that contains transect length. This is ignored if pointSurvey = TRUE.
control	A list containing optimization control parameters such as the maximum number of iterations, tolerance, the optimizer to use, etc. See the RdistanceControls function for explanation of each value, the defaults, and the requirements for this list. See examples below for how to change controls.

Details

Distances are reflected about w.lo before being passed to density. Distances exactly equal to w.lo are not reflected. Reflection around w.lo greatly improves performance of the kernel methods near the w.lo boundary where substantial non-zero probability of sighting typically exists.

Value

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

parameters	A data frame containing the $x and $y components of the smooth. $x is a vector of length 512 (default for density) evenly spaced points between w.lo and w.hi.
loglik	The value of the log likelihood. Specifically, the sum of the negative log heights of the smooth at observed distances, after the smoothed function has been scaled to integrate to one.
w.lo	Left-truncation value used during the fit.
w.hi	Right-truncation value used during the fit.
dist	The input vector of observed distances.
covars	NULL. Covariates are not allowed in the smoothed distance function (yet).
call	The original call of this function.
call.x.scl	The distance at which the distance function is scaled. This is the x at which g(x) = g.x.scl. Normally, call.x.scl = 0.
call.g.x.scl	The value of the distance function at distance call.x.scl. Normally, call.g.x.scl = 1.
call.observer	The value of input parameter observer.
fit	The smoothed object returned by stats::density. All information returned by stats::density is preserved, and in particular the numeric value of the bandwidth used during the smooth is returned in fit$bw
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.

Input data frames

To save space and to easily specify sites without detections, all site ID's, regardless whether a detection occurred there, and *site level* covariates are stored in the siteData data frame. Detection distances and group sizes are measured at the *detection level* and are stored in the detectionData data frame.

Data frame requirements: The following explains conditions under which various combinations of the input data frames are required.

1. **Detection data and site data both required:**
   Both `detectionData` and `siteData` are required if *site level* covariates are specified on the right-hand side of `formula`. *Detection level* covariates are not currently allowed.
2. **Detection data only required:**
   The `detectionData` data frame alone can be specified if no covariates are included in the distance function (i.e., right-hand side of `formula` is "~1"). Note that this routine (`dfuncEstim`) does not need to know about sites where zero targets were detected, hence `siteData` can be missing when no covariates are involved.
3. **Neither detection data nor site data required**
   Neither `detectionData` nor `siteData` are required if all variables specified in `formula` are within the scope of this routine (e.g., in the global working environment). Scoping rules here work the same as for other modeling routines in R such as `lm` and `glm`. Like other modeling routines, it is possible to mix and match the location of variables in the model. Some variables can be in the `.GlobalEnv` while others are in either `detectionData` or `siteData`.

Relationship between data frames (transect and point ID's): The input data frames, `detectionData` and `siteData`, must be merge-able on unique sites. For line-transects, site ID's (i.e., transect ID's) are unique values of the `transectID` column in `siteData`. In this case, the following merge must work: `merge(detectionData,siteData,by=transectID)`. For point-transects, site ID's (i.e., point ID's) are unique values of the combination `paste(transectID,pointID)`. In this case, the following merge must work: `merge(detectionData,siteData,by=c(transectID, pointID)`.

By default,`transectID` and `pointID` are NULL and the merge is done on all common columns. That is, when `transectID` is NULL, this routine assumes unique *transects* are specified by unique combinations of the common variables (i.e., unique values of `intersect(names(detectionData), names(siteData))`).

An error occurs if there are no common column names between `detectionData` and `siteData`. Duplicate site IDs are not allowed in `siteData`. If the same site is surveyed in multiple years, specify another transect ID column (e.g., `transectID = c("year","transectID")`). Duplicate site ID's are allowed in `detectionData`.

To help explain the relationship between data frames, bear in mind that during bootstrap estimation of variance in `abundEstim`, unique *transects* (i.e., unique values of the transect ID column(s)), not *detections* or *points*, are resampled with replacement.

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.

Scott, D. W. (1992) *Multivariate Density Estimation: Theory, Practice, and Visualization.* Wiley.

Sheather, S. J. and Jones, M. C. (1991) A reliable data-based bandwidth selection method for kernel density estimation. *Journal of the Royal Statistical Society series B*, 53, 683-690.

Silverman, B. W. (1986) *Density Estimation*. London: Chapman and Hall.

See Also

`abundEstim`, `autoDistSamp`, `dfuncEstim` for the parametric version.

Examples

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

# Compare smoothed and half-normal detection function
dfuncSmu <- dfuncSmu(dist~1, sparrowDetectionData, w.hi=units::set_units(150, "m"))
dfuncHn  <- dfuncEstim(formula=dist~1,sparrowDetectionData,w.hi=units::set_units(150, "m"))

# Print and plot results
dfuncSmu
dfuncHn
plot(dfuncSmu,main="",nbins=50)

x <- seq(0,150,length=200)
y <- dnorm(x, 0, predict(dfuncHn)[1])
y <- y/y[1]
lines(x,y, col="orange", lwd=2)
legend("topright", legend=c("Smooth","Halfnorm"),
  col=c("red","orange"), lwd=2)
```

EDR *Effective Detection Radius (EDR) for estimated detection functions with point transects*

Description

Computes Effective Detection Radius (EDR) for estimated detection functions with point transects. The point-transect equivalent to Effective Strip Width (ESW).

Usage

```
EDR(obj, newdata)
```

Arguments

obj	An estimated detection function object. An estimated detection function object has class 'dfunc', and is usually produced by a call to dfuncEstim. The estimated detection function may optionally contain a $g(0)$ component. If no $g(0)$ component is found, $g(0) = 1$ is assumed.
newdata	A data frame containing new values of the covariates at which EDR's are sought. If NULL or missing and obj contains covariates, the covariates stored in obj are used. See **Value** section.

Details

The point-transect equivalent to Effective Strip Width (ESW).

Value

If newdata is not missing and not NULL and covariates are present in obj, returned value is a vector with length equal to the number of rows in newdata. If newdata is missing or NULL and covariates are present in obj, returned value is a vector with length equal to the number of detections in obj$detections. In either of the above cases, elements in the returned vector are the effective detection radii for the corresponding set of covariates.

If obj does not contain covariates, newdata is ignored and a scalar equal to the (constant) effective detection radius for all detections is returned.

See Also

dfuncEstim, ESW, effectiveDistance

Examples

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

# Fit half-normal detection function
dfunc <- dfuncEstim(formula=dist~1
                  , detectionData=thrasherDetectionData
                  , likelihood="halfnorm"
                  , w.hi=units::set_units(175, "m")
                  , pointSurvey=TRUE)

# Compute effective detection radius (EDR)
EDR(dfunc)

# EDR only applies to point transect surveys
# ESW is the line transect equivalent
# The effectiveDistance function tests whether the dfunc was
# fit to line or point data, and returns either ESW or EDR accordingly
effectiveDistance(dfunc)
```

---

effectiveDistance *Calculates the effective sampling distance for estimated detection functions*

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(obj, newdata = NULL)
```

Arguments

obj	An estimated detection function object. An estimated detection function object has class 'dfunc', and is usually produced by a call to dfuncEstim. The estimated detection function may optionally contain a $g(0)$ component. If no $g(0)$ component is found, $g(0) = 1$ is assumed.
newdata	A data frame containing new values of the covariates at which ESW's or EDR's are sought. If NULL or missing and obj contains covariates, the covariates stored in obj are used. See **Value** section.

Details

Serves as a wrapper for ESW and EDR.

Value

If newdata is not missing or NULL and covariates are present in obj, returned value is a vector with length equal to the number of rows in newdata. If newdata is missing or NULL and covariates are present in obj, returned value is a vector with length equal to the number of detections in obj$detections. In either of the above cases, elements in the returned vector are the effective sampling distances for the corresponding set of covariates.

If obj does not contain covariates, newdata is ignored and a scalar equal to the (constant) effective sampling distance for all detections is returned.

See Also

dfuncEstim ESW EDR

---

estimateN *Abundance point estimates*

Description

Estimate abundance given a distance function, a "merged" data frame containing detections and transect lengths, area, and the number of sides surveyed (if line-transects). This is called internally by abundEstim. Most users will call abundEstim to estimate abundance.

Usage

```
estimateN(
  dfunc,
  data,
  area = NULL,
  surveyedSides,
  lengthColumn,
  control = RdistanceControls()
)
```

Arguments

dfunc	An estimate distance function (see dfuncEstim).
data	A data frame containing distance observations, transects, and lengths. This data frame must have a column named 'siteID' that identifies unique sites (transects or points). If observations were made on line-transects, this data frame must also have a column named by the lengthColumn parameter that contains transect lengths. NA length transects are accepted and are dropped when computing total transect length. Only observations on non-NA-length transects are toward density.
area	A scalar containing the total area of inference. Commonly, this is study area size. If area is NULL (the default), area will be set to 1 square unit of the output units and this produces abundance estimates equal density estimates. 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 many others are acceptable.
surveyedSides	The number of sides of the transect that were surveyed. Either 1 or 2. Only applies to line transects.
lengthColumn	Character string specifying the (single) column in siteData that contains transect lengths. This is ignored if pointSurvey = TRUE. This column must have measurement units.
control	A list containing optimization control parameters such as the maximum number of iterations, tolerance, the optimizer to use, etc. See the RdistanceControls function for explanation of each value, the defaults, and the requirements for this list. See examples below for how to change controls.

Details

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

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

where *n* is total number of sighted individuals (i.e., sum(dfunc$detections$groupSizes)), *L* is the total length of surveyed transect (i.e., sum(siteData[,lengthColumn])), 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, *P* is the total number of surveyed points, 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.

Value

A list containing the following components:

density	Estimated density in the surveyed area.
abundance	Estimated abundance on the study area.
n.groups	The number of detections (not individuals, unless all group sizes = 1) used to estimate density and abundance.
n.seen	The number of individuals (sum of group sizes) used to estimate density and abundance.
area	Total area of inference. Study area size
surveyedUnits	Number of surveyed sites. This is total transect length for line-transects and number of points for point-transects. This total transect length does not include NA transects.
surveyedSides	Number of sides (1 or 2) of transects surveyed. Only relevant for line-transects.
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, x$density is x$n.seen / (x$surveyedSides * x$w * x$pDetection * x$surveyedUnits)

See Also

dfuncEstim, abundEstim

ESW *Line transect Effective Strip Width (ESW)*

Description

Returns effective strip width (ESW) from an estimated line transect detection functions. This function applies only to line transect information. Function EDR is for point transect data. Function effectiveDistance accepts either point or line transect data.

Usage

```
ESW(obj, newdata)
```

Arguments

obj	An estimated detection function object. An estimated detection function object has class 'dfunc', and is usually produced by a call to dfuncEstim. The estimated detection function may optionally contain a $g(0)$ component that specifies detection probability on the transect. If no $g(0)$ component is found, $g(0)$ = 1 is assumed.
newdata	A data frame containing new values of the covariates at which ESW's are sought. If NULL or missing and obj contains covariates, the covariates stored in obj are used. See **Value** section.

Details

Effective strip width (ESW) of a distance function is its integral. That is, ESW is the area under the distance function from its left-truncation limit (obj$w.lo) to its right-truncation limit (obj$w.hi). In mathematical notation,

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

where $g(x)$ is the height of the distance function at distance x, and $w.lo$ and $w.hi$ are the lower and upper truncation limits used during the survey.

If detection does not decline with distance, area under the detection function is the entire half-width of the strip transect (i.e., obj$w.hi - obj$w.lo). In this case density is the number sighted targets divided by area surveyed, where area surveyed is obj$w.hi-obj$w.lo times total length of transects.

When detection declines with distance, less than the total half-width is *effectively* covered. In this case, Buckland *et al.* (1993) show that the denominator of the density estimator is total length of surveyed transects times area under the detection function (i.e., this integral). By analogy with the non-declining detection case, ESW is the transect half-width that observers *effectively* cover. In other words, if ESW = X, the study effectively covers the same area as a study with non-declining detection out to a distance of X.

A technical consideration: Rdistance uses the trapezoid rule to numerically integrate under the distance function from obj$w.lo to obj$w.hi. Two-hundred trapezoids are used in the approximation to speed calculations. In some rare cases, two hundred trapezoids may not be enough. In these cases, users should modify this function's code and bump seq.length to a value greater than 200.

Value

If newdata is not missing and not NULL and covariates are present in obj, the returned value is a vector of ESW values associated with covariates in the distance function and equal in length to the number of rows in newdata. If newdata is missing or NULL and covariates are present in obj, an ESW vector with length equal to the number of detections in obj$detections is returned.

If obj does not contain covariates, newdata is ignored and a scalar equal to the (constant) effective strip width for all detections is returned.

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, EDR, effectiveDistance

Examples

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

dfunc <- dfuncEstim(formula=dist~1
                  , detectionData = sparrowDetectionData)

# Compute effective strip width (ESW)
ESW(dfunc)
```

`F.double.obs.prob` *Compute double observer probability of detection (No external covariates allowed)*

Description

Estimates the probability of detection in a two-observer system when observations are independent.

Usage

```
F.double.obs.prob(df, observer = "both")
```

Arguments

`df`	A data frame containing the components `$obsby.1` and `$obsby.2`. These components are either 0/1 (0 = missed, 1 = seen) or TRUE/FALSE (logical) vectors indicating whether observer 1 (`obsby.1`) or observer 2 (`obsby.2`) spotted the target. There is no flexibility on naming these columns of `df`. They must be named `$obsby.1` and `$obsby.2`.
`observer`	A number of text string indicating the primary observer. Primary observers can be observer 1, or observer 2, or "both". If, for example, observer 2 was a data recorder and part-time observer, or if observer 2 was the pilot, set `observer = 1`. This dictates which set of observations form the denominator of the double observer system. For example, if `observer = 1`, observations by observer 1 that were not seen by observer 2 are ignored. The estimate in this case uses targets seen by both observers and those seen by observer 2 but not observer 1. If observer = "both", the denominator is computed twice, once assuming observer 1 was the primary, once assuming observer 2 was the primary, and then computes the probability of one or more observers sighting a target.

Details

When observer = "both", the observers are assumed to be independent. In this case the estimate of detection is

$$p = p_1 + p_2 - p_1 p_2$$

where p_1 is the proportion of targets seen by observer 2 that were also seen by observer 1, p_2 is the proportion of targets seen by observer 1 that were also seen by observer 2. This estimator is very close to unbiased when observers are actually independent.

Value

A single scalar, the probability of detection estimate.

See Also

dfuncEstim, abundEstim

Examples

```
#   Fake observers
  set.seed(538392)
  obsrv <- data.frame( obsby.1=rbinom(100,1,.75), obsby.2=rbinom(100,1,.5) )

  F.double.obs.prob( obsrv, observer=1 )
  F.double.obs.prob( obsrv, observer=2 )
  F.double.obs.prob( obsrv, observer="both" )
```

F.gx.estim *F.gx.estim - Estimate g(0) or g(x)*

Description

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

Usage

```
F.gx.estim(fit, x.scl = NULL, g.x.scl = NULL, observer = NULL)
```

Arguments

fit	An estimated dfunc object. See dfuncEstim.
x.scl	The x coordinate (a distance) at which to scale the sightability function to g.x.scl, 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. See Details for more on scaling the sightability function.

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. See Details.

observer — A numeric scalar or text string specifying whether observer 1 or observer 2 or both were full-time observers. This parameter dictates which set of observations form the denominator of a double observer system. If, for example, observer 2 was a data recorder and part-time observer, or if observer 2 was the pilot, set observer = 1. If observer = 1, observations by observer 1 not seen by observer 2 are ignored. The estimate of detection in this case is the ratio of number of targets seen by both observers to the number seen by both plus the number seen by just observer 2. If observer = "both", the computation goes both directions.

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. This case is the default, with g.x.scl = 1, when gamma distance functions are estimated.
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

```
set.seed(555574)
x <- rnorm(1000) * 100
x <- x[ 0 < x & x < 100 ]
x <- units::set_units(x, "m")
un.dfunc <- dfuncEstim( x ~ 1
                      , likelihood = "logistic")
F.gx.estim(un.dfunc)

x <- rgamma(1000, shape = 5)
x <- units::set_units(x, "m")
gam.dfunc <- dfuncEstim( x ~ 1
                       , likelihood="Gamma")
F.gx.estim(gam.dfunc)
```

F.maximize.g *Find the coordinate of the maximum of a distance function*

Description

Find the x coordinate that maximizes g(x).

Usage

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

Arguments

fit	An estimated 'dfunc' object produced by dfuncEstim.
covars	Covariate values to calculate maximum for.

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" )

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

## End(Not run)
```

`F.nLL` *Return the negative log likelihood for a set of distance values*

Description

Return value of the negative log likelihood for a vector of observed distances given a specified likelihood, number of expansion terms, and estimated parameters.

Usage

```
F.nLL(
  a,
  dist,
  covars = NULL,
  like,
  w.lo = 0,
  w.hi = max(dist),
  series,
  expansions = 0,
  pointSurvey,
  for.optim = F
)
```

Arguments

a	A vector of parameter values for the likelihood. Length of this vector must be expansions + 1 + 1*(like %in% c("hazrate", "uniform")).
dist	A vector of observed distances. All values must be between w.lo and w.hi (see below).
covars	Data frame containing values of covariates at each observation in dist.
like	String specifying the form of the likelihood. Built-in distance functions at present are "uniform", "halfnorm", "hazrate", "negexp", and "Gamma". To be valid, a function named paste(like,".like") (e.g., "uniform.like") must exist somewhere in this routine's scope. This routine finds the ".like" function and calls it with the appropriate parameters. A user-defined likelihood can be implemented by simply defining a function with the ".like" extension and giving the root name here. For example, define a function named "myLike.like" in the .GlobalEnv and set like="myLike" here. See the vignette on this topic.
w.lo	Lower or left-truncation limit of the distances. This is the minimum possible off-transect distance. Default is 0.
w.hi	Upper or right-truncation limit of the distances. This is the maximum off-transect distance that could be observed. Default is the maximum observed distance.
series	String specifying the type of expansion to use series if expansions > 0. Valid values at present are 'simple', 'hermite', and 'cosine'.
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.
pointSurvey	Boolean. TRUE if dist is point transect data, FALSE if line transect data.
for.optim	Boolean. If TRUE, values are multiplied by 10^9 to help optim converge more consistently.

Value

A scalar, the negative of the log likelihood evaluated at parameters a, including expansion terms.

See Also

See uniform.like and links there; dfuncEstim

F.start.limits *Set starting values and limits for parameters of Rdistance functions*

Description

Return reasonable starting values and limits (boundaries) for the parameters of distance functions. Starting values and limits are specified for all likelihoods and expansion terms. This function is called by other routines in Rdistance, and is not intended to be called by the user.

Usage

```
F.start.limits(
  like,
  expan,
  w.lo,
  w.hi,
  dist,
  covars = NULL,
  pointSurvey = FALSE
)
```

Arguments

like	String specifying the likelihood for the distance function. Possible values are "hazrate" for hazard rate likelihood, "halfnorm" for the half normal likelihood, "uniform" for the uniform likelihood, "negexp" for the negative exponential likelihood, and "Gamma" for the gamma likelihood.
expan	Number of expansion terms to include. Valid values are 0, 1, ..., 3.
w.lo	Lower or left-truncation limit of the distances. Normally, 0.
w.hi	Upper or right-truncation limit of the distances. This is the maximum off-transect distance that could be observed.
dist	The vector of observed off-transect distances being analyzed. This vector is only required for like = "Gamma" and "halfnorm".
covars	Matrix of covariate values.
pointSurvey	Boolean. TRUE if point transect data, FALSE if line transect data.

Details

The number of parameters to be fitted is expan + 1 + 1*(like %in% c("hazrate", "uniform")). This is the length of all vectors returned in the output list.

Value

A list containing the following components

start	Vector of reasonable 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.

See Also

dfuncEstim

Examples

```
data(sparrowDetectionData)
dist <- sparrowDetectionData$dist
units(dist) <- "m"
wl <- units::as_units(0, "m")
wh <- units::as_units(1000, "m")

F.start.limits("uniform", 0, wl, wh, dist)
F.start.limits("uniform", 1, wl, wh, dist)
F.start.limits("uniform", 2, wl, wh, dist)
F.start.limits("uniform", 3, wl, wh, dist)

F.start.limits("halfnorm", 0, wl, wh, dist)
F.start.limits("halfnorm", 1, wl, wh, dist)
F.start.limits("halfnorm", 2, wl, wh, dist)
F.start.limits("halfnorm", 3, wl, wh, dist)

F.start.limits("halfnorm", 0, wl, wh, dist, pointSurvey = TRUE)
F.start.limits("halfnorm", 1, wl, wh, dist, pointSurvey = TRUE)
F.start.limits("halfnorm", 2, wl, wh, dist, pointSurvey = TRUE)
F.start.limits("halfnorm", 3, wl, wh, dist, pointSurvey = TRUE)

F.start.limits("halfnorm", 0, wl, wh, dist, data.frame(A=1, B=2))
F.start.limits("halfnorm", 1, wl, wh, dist, data.frame(A=1, B=2))
F.start.limits("halfnorm", 2, wl, wh, dist, data.frame(A=1, B=2))
F.start.limits("halfnorm", 3, wl, wh, dist, data.frame(A=1, B=2))

F.start.limits("halfnorm", 0, wl, wh, dist, data.frame(A=1, B=2), TRUE)
F.start.limits("halfnorm", 1, wl, wh, dist, data.frame(A=1, B=2), TRUE)
F.start.limits("halfnorm", 2, wl, wh, dist, data.frame(A=1, B=2), TRUE)
F.start.limits("halfnorm", 3, wl, wh, dist, data.frame(A=1, B=2), TRUE)

F.start.limits("hazrate", 0, wl, wh, dist)
F.start.limits("hazrate", 1, wl, wh, dist)
F.start.limits("hazrate", 2, wl, wh, dist)
F.start.limits("hazrate", 3, wl, wh, dist)

F.start.limits("negexp", 0, wl, wh, dist)
F.start.limits("negexp", 1, wl, wh, dist)
F.start.limits("negexp", 2, wl, wh, dist)
F.start.limits("negexp", 3, wl, wh, dist)

F.start.limits("Gamma", 0, wl, wh, dist)
```

`Gamma.like` *Gamma.like - Gamma distance function*

Description

Computes the gamma likelihood, scaled appropriately, for use as a likelihood in estimating a distance function.

Usage

```
Gamma.like(
  a,
  dist,
  covars = NULL,
  w.lo = units::set_units(0, "m"),
  w.hi = max(dist),
  series = "cosine",
  expansions = 0,
  scale = TRUE,
  pointSurvey = FALSE
)
```

Arguments

a	A vector of likelihood parameter values. Length and meaning depend on series and expansions. If no expansion terms were called for (i.e., expansions = 0), the distance likelihoods 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. If p is the number of canonical parameters, coefficients for the expansion terms are a[(p+1):length(a)].
dist	A numeric vector containing the observed distances.
covars	Data frame containing values of covariates at each observation in dist.
w.lo	Scalar value of the lowest observable distance. This is the *left truncation* of sighting distances in dist. Same units as dist. Values less than w.lo are allowed in dist, but are ignored and their contribution to the likelihood is set to NA in the output.
w.hi	Scalar value of the largest observable distance. This is the *right truncation* of sighting distances in dist. Same units as dist. Values greater than w.hi are allowed in dist, but are ignored and their contribution to the likelihood is set to NA in the output.
series	A string specifying the type of expansion to use. Currently, valid values are 'simple', 'hermite', and 'cosine'; but, see dfuncEstim about defining other series.
expansions	A scalar specifying the number of terms in series. Depending on the series, this could be 0 through 5. The default of 0 equates to no expansion terms of any type.
scale	Logical scalar indicating whether or not to scale the likelihood so it integrates to 1. This parameter is used to stop recursion in other functions. If scale equals TRUE, a numerical integration routine (integration.constant) is called, which in turn calls this likelihood function again with scale = FALSE. Thus, this routine knows when its values are being used to compute the likelihood and when its value is being used to compute the constant of integration. All user defined likelihoods must have and use this parameter.
pointSurvey	Boolean. TRUE if dist is point transect data, FALSE if line transect data.

Details

This function utilizes the built-in R function dgamma to evaluate the gamma density function. Using the parameterization of dgamma, the gamma shape parameter is a[1] while the gamma scale parameter is (a[2]/gamma(r)) * (((r - 1)/exp(1))^(r - 1)). Currently, this function implements a non-covariate version of the gamma detection function used by Becker and Quang (2009). In future, linear equations will relate covariate values to values of the gamma parameters. This future implementation will fully replicate the distance functions of Becker and Quang (2009).

Value

A numeric vector the same length and order as dist containing the likelihood contribution for distances in dist. Assuming L=gamma.like(c(r,lam),dist), the full log likelihood of all the data is -sum(log(L), na.rm=T). Note that the returned likelihood value for distances less than w.lo or greater than w.hi is NA, and thus it is prudent to use na.rm=TRUE in the sum. If scale = TRUE, the integral of the likelihood from w.lo to w.hi is 1.0. If scale = FALSE, the integral of the likelihood is an arbitrary constant.

References

Becker, E. F., and P. X. Quang, 2009. *A Gamma-Shaped Detection Function for Line-Transect Surveys with Mark-Recapture and Covariate Data.* Journal of Agricultural, Biological, and Environmental Statistics 14(2):207-223.

See Also

dfuncEstim, halfnorm.like, hazrate.like, uniform.like, negexp.like

Examples

```
## Not run:
set.seed(238642)
x <- seq(0, 100, length=100)

# Plots showing effects of changes in shape
plot(x, Gamma.like(c(20,20), x), type="l", col="red")
plot(x, Gamma.like(c(40,20), x), type="l", col="blue")

# Plots showing effects of changes in scale
plot(x, Gamma.like(c(20,20), x), type="l", col="red")
plot(x, Gamma.like(c(20,40), x), type="l", col="blue")

# Estimate 'Gamma' distance function
r <- 5
lam <- 10
b <- (1/gamma(r)) * (((r - 1)/exp(1))^(r - 1))
x <- rgamma(1000, shape=r, scale=b*lam)
dfunc <- dfuncEstim(x~1, likelihood="Gamma", x.scl="max")
plot(dfunc)

## End(Not run)
```

Gamma.start.limits *Gamma.start.limits - Start and limit values for Gamma parameters.*

Description

Compute starting values and limits for the Gamma likelihood function.

Usage

```
Gamma.start.limits(dist, covars, expansions, w.lo, w.hi)
```

Arguments

dist	A numeric vector containing observed distances with measurement units.
covars	Data frame containing values of covariates at each observation in dist.
expansions	A scalar specifying the number of terms in series. Depending on the series, this could be 0 through 5. The default of 0 equates to no expansion terms of any type.
w.lo	Scalar value of the lowest observable distance, with measurement units. This is the *left truncation* sighting distance. Values less than w.lo are allowed in dist, but are ignored and their likelihood value is set to NA in the output.
w.hi	Scalar value of the largest observable distance, with measurement units. This is the *right truncation* sighting distance. Values greater than w.hi are allowed in dist, but are ignored and their likelihood value is set to NA in the output.

Value

A list containing the following components:

- start : a vector of starting values
- lowlimit : a vector of lower limits (can be -Inf)
- highlimit : a vector of upper limits (can be Inf)
- nms : a vector containing names of the parameters

getDfuncModelFrame *Return model frame for dfunc*

Description

Returns the model frame from a formula and data set. This routine is intended to only be called from within other Rdistance functions.

Usage

```
getDfuncModelFrame(formula, data)
```

Arguments

formula	A dfunc formula object. See dfuncEstim.
data	The data frame from which variables in formula (potentially) come.

Details

This routine is needed to get the scoping correct in dfuncEstim. In dfuncEstim, we first merge the detection and site data frames, then call this routine.

Value

a model frame containing the response and covariates resulting from evaluating formula in data.

---

halfnorm.like *Half-normal likelihood function for distance analyses*

Description

This function computes the likelihood contributions for sighting distances, scaled appropriately, for use as a distance likelihood.

Usage

```
halfnorm.like(
  a,
  dist,
  covars = NULL,
  w.lo = units::set_units(0, "m"),
  w.hi = max(dist),
  series = "cosine",
  expansions = 0,
  scale = TRUE,
  pointSurvey = FALSE
)
```

Arguments

a	A vector of likelihood parameter values. Length and meaning depend on series and expansions. If no expansion terms were called for (i.e., expansions = 0), the distance likelihoods 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., if p is the number of canonical parameters, coefficients for the expansion terms are a[(p+1):length(a)].

`dist`	A numeric vector containing the observed distances.
`covars`	Data frame containing values of covariates at each observation in `dist`.
`w.lo`	Scalar value of the lowest observable distance. This is the *left truncation* of sighting distances in `dist`. Same units as `dist`. Values less than `w.lo` are allowed in `dist`, but are ignored and their contribution to the likelihood is set to `NA` in the output.
`w.hi`	Scalar value of the largest observable distance. This is the *right truncation* of sighting distances in `dist`. Same units as `dist`. Values greater than `w.hi` are allowed in `dist`, but are ignored and their contribution to the likelihood is set to `NA` in the output.
`series`	A string specifying the type of expansion to use. Currently, valid values are 'simple', 'hermite', and 'cosine'; but, see `dfuncEstim` about defining other series.
`expansions`	A scalar specifying the number of terms in `series`. Depending on the series, this could be 0 through 5. The default of 0 equates to no expansion terms of any type.
`scale`	Logical scalar indicating whether or not to scale the likelihood so it integrates to 1. This parameter is used to stop recursion in other functions. If `scale` equals TRUE, a numerical integration routine (`integration.constant`) is called, which in turn calls this likelihood function again with `scale` = FALSE. Thus, this routine knows when its values are being used to compute the likelihood and when its value is being used to compute the constant of integration. All user defined likelihoods must have and use this parameter.
`pointSurvey`	Boolean. TRUE if distances in `dist` are radial from point transects, FALSE if distances are perpendicular off-transect distances.

Details

The half-normal likelihood is

$$f(x|a) = \exp(-x^2/(2 * a^2))$$

where a is the parameter to be estimated. Some half-normal distance functions in the literature do not use a "2" in the denominator of the exponent. `Rdistance` uses a "2" in the denominator of the exponent to make quantiles of this function agree with the standard normal which means a can be interpreted as a normal standard error. e.g., approximately 95% of all observations will occur between 0 and $2a$.

Expansion Terms: If `expansions` = k (k > 0), the expansion function specified by `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,

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

f(x|a,b,c_1,c_2,...,c_k) = f(x|a,b)(1 + c(1) h_i1(x) + c(2) h_i2(x) + ... + c(k) h_ik(x)).

Value

A numeric vector the same length and order as `dist` containing the likelihood contribution for corresponding distances in `dist`. Assuming `L` is the returned vector from one of these functions, the negative log likelihood of all the data is `-sum(log(L), na.rm=T)`. Note that the returned likelihood value for distances less than `w.lo` or greater than `w.hi` is `NA`, hence `na.rm=TRUE` in the sum. If `scale` = TRUE, the integral of the likelihood from `w.lo` to `w.hi` is 1.0. If `scale` = FALSE, the integral of the likelihood is something else. Values are always greater than or equal to zero.

See Also

`dfuncEstim`, `hazrate.like`, `uniform.like`, `negexp.like`, `Gamma.like`

Examples

```
 ## Not run:
set.seed(238642)
x <- seq(0, 100, length=100)

# Plots showing effects of changes in parameter Sigma
plot(x, halfnorm.like(20, x), type="l", col="red")
plot(x, halfnorm.like(40, x), type="l", col="blue")

# Estimate 'halfnorm' distance function
a <- 5
x <- rnorm(1000, mean=0, sd=a)
x <- x[x >= 0]
dfunc <- dfuncEstim(x~1, likelihood="halfnorm")
plot(dfunc)

# evaluate the log Likelihood
L <- halfnorm.like(dfunc$parameters, dfunc$detections$dist, covars=dfunc$covars,
    w.lo=dfunc$w.lo, w.hi=dfunc$w.hi,
    series=dfunc$series, expansions=dfunc$expansions,
    scale=TRUE)
-sum(log(L), na.rm=TRUE)  # the negative log likelihood

## End(Not run)
```

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

Description

Computes the hazard rate likelihood of off-transect distances, given parameters. Primarily used as a minimization objective during distance function estimation.

Usage

```
hazrate.like(
  a,
  dist,
  covars = NULL,
  w.lo = units::set_units(0, "m"),
  w.hi = max(dist),
  series = "cosine",
  expansions = 0,
  scale = TRUE,
  pointSurvey = FALSE
)
```

Arguments

a	A vector of likelihood parameter values. Length and meaning depend on series and expansions. If no expansion terms were called for (i.e., expansions = 0), the distance likelihoods 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. If p is the number of canonical parameters, coefficients for the expansion terms are a[(p+1):length(a)].
dist	A numeric vector containing the observed distances.
covars	Data frame containing values of covariates at each observation in dist.
w.lo	Scalar value of the lowest observable distance. This is the *left truncation* of sighting distances in dist. Same units as dist. Values less than w.lo are allowed in dist, but are ignored and their contribution to the likelihood is set to NA in the output.
w.hi	Scalar value of the largest observable distance. This is the *right truncation* of sighting distances in dist. Same units as dist. Values greater than w.hi are allowed in dist, but are ignored and their contribution to the likelihood is set to NA in the output.
series	A string specifying the type of expansion to use. Currently, valid values are 'simple', 'hermite', and 'cosine'; but, see dfuncEstim about defining other series.
expansions	A scalar specifying the number of terms in series. Depending on the series, this could be 0 through 5. The default of 0 equates to no expansion terms of any type.
scale	Logical scalar indicating whether or not to scale the likelihood so it integrates to 1. This parameter is used to stop recursion in other functions. If scale equals TRUE, a numerical integration routine (integration.constant) is called, which in turn calls this likelihood function again with scale = FALSE. Thus, this routine knows when its values are being used to compute the likelihood and when its value is being used to compute the constant of integration. All user defined likelihoods must have and use this parameter.
pointSurvey	Boolean. TRUE if dist is point transect data, FALSE if line transect data.

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.

Expansion Terms: If `expansions` = e (e > 0), the expansion function specified by `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,

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

Value

A numeric vector the same length and order as `dist` containing the likelihood contribution for corresponding distances in `dist`. Assuming `L` is the returned vector from one of these functions, the full log likelihood of all the data is `-sum(log(L), na.rm=T)`. Note that the returned likelihood value for distances less than `w.lo` or greater than `w.hi` is `NA`, and thus it is prudent to use `na.rm=TRUE` in the sum. If `scale` = TRUE, the integral of the likelihood from `w.lo` to `w.hi` is 1.0. If `scale` = FALSE, the integral of the likelihood is arbitrary.

See Also

`dfuncEstim`, `halfnorm.like`, `uniform.like`, `negexp.like`, `Gamma.like`

Examples

```
## Not run:
x <- seq(0, 100, length=100)

# Plots showing effects of changes in sigma
plot(x, hazrate.like(c(20, 5), x), type="l", col="red")
plot(x, hazrate.like(c(40, 5), x), type="l", col="blue")

# Plots showing effects of changes in beta
plot(x, hazrate.like(c(50, 20), x), type="l", col="red")
plot(x, hazrate.like(c(50, 2), x), type="l", col="blue")

## End(Not run)
```

`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

```
set.seed(83828233)
  x <- rnorm(1000) * 100
  x <- x[0 < x & x < 100]
  herm.expn <- hermite.expansion(x, 3)
```

---

integration.constant *Compute the integration constant for distance density functions*

Description

Using numerical integration, this function computes the area under a distance function between two limits (w.lo and w.hi).

Usage

```
integration.constant(
  dist,
  density,
  a,
  covars,
  w.lo,
  w.hi,
  series,
  expansions,
  pointSurvey
)
```

Arguments

dist	Vector of detection distance values.
density	A likelihood function for which the integration constant is sought. This function must be capable of evaluating values between w.lo and w.hi and have the following parameters: • 'a' = Parameter vector. • 'dist' = Vector of distances. • 'covars' = If the density allows covariates, the covariate matrix. • 'w.lo' = Lower limit or left truncation value. • 'w.hi' = Upper limit or right truncation value. • 'series' = Form of the series expansions, if any. • 'expansions' = Number of expansion terms.

- 'scale' = Whether to scale function to integrate to 1.

a	Vector of parameters to pass to density.
covars	Matrix of covariate values.
w.lo	The lower limit of integration, or the left truncation value for perpendicular distances.
w.hi	The upper limit of integration, or the right truncation value for perpendicular distances.
series	The series to use for expansions. If expansions > 0, this string specifies the type of expansion. Valid values at present are 'simple', 'hermite', and 'cosine'.
expansions	Number of expansions in density.
pointSurvey	Boolean. TRUE if point transect data, FALSE if line transect data.

Details

The trapezoid rule is used to numerically integrate density from w.lo to w.hi. Two-hundred (200) equal-sized trapezoids are used in the integration. The number of trapezoids to use is fixed and cannot be changed without re-writing this routine.

Value

A scalar (or vector of scalars if covariates are present) that is the area under density between w.lo and w.hi. This scalar can be used as a divisor to scale density such that it integrates to 1.0. If x = density(...), then x / integration.constant(density, ...) will integrate to 1.0.

See Also

dfuncEstim, halfnorm.like

Examples

```
# Can put any number for first argument (1 used here)
scl <- integration.constant(dist=units::set_units(1,"m")
                          , density=logistic.like
                          , covars = NULL
                          , pointSurvey = FALSE
                          , w.lo = units::set_units(0,"m")
                          , w.hi = units::set_units(100,"m")
                          , expansions = 0
                          , a=c(75,25))
print(scl) # Should be 75.1

x <- units::set_units(seq(0,100,length=200), "m")
y <- logistic.like( c(75,25), x, scale=FALSE ) / scl
int.y <- (x[2]-x[1]) * sum(y[-length(y)]+y[-1]) / 2  # the trapezoid rule, should be 1.0
print(int.y) # Should be 1
```

isUnitless *isUnitless - 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

```
isUnitless(obj)
```

Arguments

`obj`	A numeric scaler 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
isUnitless(a)
isUnitless(b)
isUnitless(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 - Lines method for distance (detection) functions*

Description

Lines method for objects of class 'dfunc'. Distance function line methods add distance functions to existing plots.

Usage

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

Arguments

x	An estimated distance function resulting from a call to dfuncEstim.
newdata	Data frame similar to the newdata parameter to 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, the routine computes the mean of all numeric covariates in the distance function and the mode of all factor covariates in the distance function. The new mean and mode vector is used to predict and plot a distance function.
...	Parameters to lines used to 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.

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")
dfunc <- dfuncEstim(x~1, likelihood="halfnorm")
plot(dfunc, nbins = 40, col="lightgrey", border=NA, vertLines=FALSE)
lines(dfunc, col="grey", lwd=15)
lines(dfunc, col="black", lwd=5, lty = 2)

# Multiple lines
data(sparrowDetectionData)
data(sparrowSiteData)
dfuncObs <- dfuncEstim(formula = dist ~ observer
                     , likelihood = "halfnorm"
                     , detectionData = sparrowDetectionData
                     , siteData = sparrowSiteData)
plot(dfuncObs
   , vertLines = FALSE
   , lty = 0
   , col = c("grey","lightgrey")
   , border=NA
   , main="Detection by observer"
   , legend = FALSE)
y <- lines(dfuncObs
   , newdata = data.frame(observer = levels(sparrowSiteData$observer))
   , col = palette.colors(length(levels(sparrowSiteData$observer)))
   , lty = 1
   , lwd = 4)
head(y) # values returned, same as predict method
```

---

logistic.like *logistic.like - Logistic distance function likelihood*

Description

Computes a two parameter logistic distance function.

Usage

```
logistic.like(
  a,
  dist,
  covars = NULL,
  w.lo = units::set_units(0, "m"),
  w.hi = max(dist),
  series = "cosine",
  expansions = 0,
```

```
  scale = TRUE,
  pointSurvey = FALSE
)
```

Arguments

a	A vector of likelihood parameter values. Length and meaning depend on whether covariates and expansions are present as follows: • If no covariates and no expansions: a = [a, b] (see Details) • If no covariates and *k* expansions: a = [a, b, e1, ..., e*k*] • If *p* covariates and no expansions: a = [a, b, b1, ..., b*p*] • If *p* covariates and *k* expansions: a = [a, b, b1, ..., b*p*, e1, ..., e*k*]
dist	A numeric vector containing observed distances with measurement units.
covars	Data frame containing values of covariates at each observation in dist.
w.lo	Scalar value of the lowest observable distance, with measurement units. This is the *left truncation* sighting distance. Values less than w.lo are allowed in dist, but are ignored and their likelihood value is set to NA in the output.
w.hi	Scalar value of the largest observable distance, with measurement units. This is the *right truncation* sighting distance. Values greater than w.hi are allowed in dist, but are ignored and their likelihood value is set to NA in the output.
series	A string specifying the type of expansion to use. Currently, valid values are 'simple', 'hermite', and 'cosine'; but, see dfuncEstim about defining other series.
expansions	A scalar specifying the number of terms in series. Depending on the series, this could be 0 through 5. The default of 0 equates to no expansion terms of any type.
scale	Logical scalar indicating whether or not to scale the likelihood into a density function, i.e., so that it integrates to 1. This parameter is used to stop recursion in other functions. If scale equals TRUE, a numerical integration routine (integration.constant) is called, which in turn calls this likelihood function again with scale = FALSE. Thus, this routine knows when its values are being used to compute the likelihood and when its values are being used to compute the constant of integration. All user defined likelihoods must have and use this parameter.
pointSurvey	Boolean. TRUE if dist is point transect data, FALSE if line transect data.

Details

The 'logistic' likelihood contains two parameters. Parameter a determines the scale and is labeled 'threshold' in Rdistance. Parameter b determines sharpness (slope) of the likelihood's decrease at a and is labeled 'knee' in Rdistance. This function is sometimes called the *heavy side* function (e.g., engineering). The technical form of the function is,

$$f(x|a,b) = 1 - \frac{1}{1+\exp(-b(x-a))} = \frac{\exp(-b(x-a))}{1+exp(-b(x-a))},$$

Parameter a is the location (distance) of 0.5. That is, the inverse likelihood of 0.5 is a before scaling (i.e., `logistic.like( c(a,b), a, scale=FALSE)` equals `0.5`).

Parameter b is slope of function at a. Prior to scaling, slope of the likelihood at a is $-b/4$. If b is large, the "knee" is sharp and the likelihood can look uniform with support from `w.lo` to $a/f(0)$. If b is small, the "knee" is shallow and the density of observations declines in an elongated "S" shape pivoting at $a/f(0)$. As b grows large and assuming f(0) = 1, the effective strip width approaches a.

See plots in Examples.

Value

A numeric vector the same length and order as `dist` containing the likelihood contribution for corresponding distances in `dist`. Assuming `L` is the returned vector, the log likelihood of all data is `-sum(log(L), na.rm=T)`. Note that the returned likelihood value for distances less than `w.lo` or greater than `w.hi` is `NA`, and thus it is essential to use `na.rm=TRUE` in the sum. If `scale` = TRUE, the integral of the likelihood from `w.lo` to `w.hi` is 1.0. If `scale` = FALSE, the integral of the likelihood is arbitrary.

Expansion Terms

If `expansions` = k (k > 0), the expansion function specified by `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, the likelihood contribution for the i^{th} distance is,

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

See Also

`dfuncEstim`, `halfnorm.like`, `hazrate.like`, `negexp.like`, `Gamma.like`

Examples

```
x <- units::set_units(seq(0, 100, length=100), "m")

# Plots showing effects of changes in Threshold
plot(x, logistic.like(c(20, 20), x), type="l", col="red")
lines(x, logistic.like(c(40, 20), x), type="l", col="blue")

# Plots showing effects of changes in Knee
plot(x, logistic.like(c(50, 100), x), type="l", col="red")
lines(x, logistic.like(c(50, 1), x), type="l", col="blue")
lines(x, logistic.like(c(50, .1), x), type="l", col="orange")
```

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

Description

Starting values and limits for parameters of the logistic distance function.

Usage

```
logistic.start.limits(dist, covars, expansions, w.lo, w.hi)
```

Arguments

`dist`	A numeric vector containing observed distances with measurement units.
`covars`	Data frame containing values of covariates at each observation in `dist`.
`expansions`	A scalar specifying the number of terms in `series`. Depending on the series, this could be 0 through 5. The default of 0 equates to no expansion terms of any type.
`w.lo`	Scalar value of the lowest observable distance, with measurement units. This is the *left truncation* sighting distance. Values less than `w.lo` are allowed in `dist`, but are ignored and their likelihood value is set to `NA` in the output.
`w.hi`	Scalar value of the largest observable distance, with measurement units. This is the *right truncation* sighting distance. Values greater than `w.hi` are allowed in `dist`, but are ignored and their likelihood value is set to `NA` in the output.

Details

This function is usually called within `F.start.limits`.

Value

A list containing the following components:

- `start` : a vector of starting values
- `lowlimit` : a vector of lower limits (can be -Inf)
- `highlimit` : a vector of upper limits (can be Inf)
- `nms` : a vector containing names of the parameters

negexp.like *negexp.like - Negative exponential distance function*

Description

Computes the negative exponential form of a distance function

Usage

```
negexp.like(
  a,
  dist,
  covars = NULL,
  w.lo = units::set_units(0, "m"),
  w.hi = max(dist),
  series = "cosine",
  expansions = 0,
  scale = TRUE,
  pointSurvey = FALSE
)
```

Arguments

a	A vector of likelihood parameter values. Length and meaning depend on series and expansions. If no expansion terms were called for (i.e., expansions = 0), the distance likelihood contains only one canonical parameter, which is the first element of a (see Details). If one or more expansions are called for, coefficients for the expansion terms follow coefficients for the canonical parameter.
dist	A numeric vector containing the observed distances.
covars	Data frame containing values of covariates at each observation in dist.
w.lo	Scalar value of the lowest observable distance. This is the *left truncation* of sighting distances in dist. Same units as dist. Values less than w.lo are allowed in dist, but are ignored and their contribution to the likelihood is set to NA in the output.
w.hi	Scalar value of the largest observable distance. This is the *right truncation* of sighting distances in dist. Same units as dist. Values greater than w.hi are allowed in dist, but are ignored and their contribution to the likelihood is set to NA in the output.
series	A string specifying the type of expansion to use. Currently, valid values are 'simple', 'hermite', and 'cosine'; but, see dfuncEstim about defining other series.
expansions	A scalar specifying the number of terms in series. Depending on the series, this could be 0 through 5. The default of 0 equates to no expansion terms of any type.

scale	Logical scalar indicating whether or not to scale the likelihood so it integrates to 1. This parameter is used to stop recursion in other functions. If scale equals TRUE, a numerical integration routine (integration.constant) is called, which in turn calls this likelihood function again with scale = FALSE. Thus, this routine knows when its values are being used to compute the likelihood and when its value is being used to compute the constant of integration. All user defined likelihoods must have and use this parameter.
pointSurvey	Boolean. TRUE if dist is point transect data, FALSE if line transect data.

Details

The negative exponential likelihood is

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

where a is a slope parameter to be estimated.

Expansion Terms: If the number of expansions = k (k > 0), the expansion function specified by 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 numeric vector the same length and order as dist containing the likelihood contribution for corresponding distances in dist. Assuming L is the returned vector from one of these functions, the full log likelihood of all the data is -sum(log(L), na.rm=T). Note that the returned likelihood value for distances less than w.lo or greater than w.hi is NA, and thus it is prudent to use na.rm=TRUE in the sum. If scale = TRUE, the integral of the likelihood from w.lo to w.hi is 1.0. If scale = FALSE, the integral of the likelihood is arbitrary.

See Also

dfuncEstim, halfnorm.like, uniform.like, hazrate.like, Gamma.like

Examples

```
## Not run:
set.seed(238642)
x <- seq(0, 100, length=100)

# Plots showing effects of changes in parameter Beta
plot(x, negexp.like(0.01, x), type="l", col="red")
plot(x, negexp.like(0.05, x), type="l", col="blue")

# Estimate 'negexp' distance function
Beta <- 0.01
x <- rexp(1000, rate=Beta)
```

```
dfunc <- dfuncEstim(x~1, likelihood="negexp")
plot(dfunc)

## End(Not run)
```

---

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 F.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,
  include.zero = FALSE,
  nbins = "Sturges",
  newdata = NULL,
  legend = TRUE,
  vertLines = TRUE,
  plotBars = TRUE,
  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 distance function resulting from a call to 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 input distance range (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 the newdata parameter to 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, the routine computes the mean of all numeric covariates in the distance function and the mode of all factor covariates in the distance function. The new mean and mode vector is used to predict and plot a distance function.

legend — Logical scalar for whether to include a legend. If TRUE, a legend will be included on the plot detailing the covariate values use 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).

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 is >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 set up the plotting region and plot bars. When bars are not plotted, this routine sets up the plot with `graphics::plot`. `...` can be any other argument to `barplot` or `plot` EXCEPT `width`, `ylim`, `xlim`, `density`, `angle`, and `space`.

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 scaler 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")
dfunc <- dfuncEstim(x~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="wheat"
   , 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(dfunc$detections$dist)

# Plot showing f(0)
hist(dfunc$detections$dist
   , n = 40
   , border = NA
   , prob = TRUE)
x <- seq(dfunc$w.lo, dfunc$w.hi, length=200)
y <- predict(dfunc, type="dfunc", distances = x)
lines(x, c(y)/attr(y, "scaler"))
c(attr(y,"scaler") / y[1], ESW(dfunc))  # 1/f(0) = ESW

# Covariates: detection by observer
data(sparrowDetectionData)
data(sparrowSiteData)
dfuncObs <- dfuncEstim(formula = dist ~ observer + groupsize(groupsize)
                     , likelihood = "hazrate"
                     , detectionData = sparrowDetectionData
                     , siteData = sparrowSiteData)
plot(dfuncObs
   , newdata = data.frame(observer = levels(sparrowSiteData$observer))
   , vertLines = FALSE
   , lty = c(1,1)
   , col.dfunc = heat.colors(length(levels(sparrowSiteData$observer)))
   , col = c("grey","lightgrey")
```

```
, border=NA
, main="Detection by observer")
```

`predict.dfunc` *Predict method for dfunc objects*

Description

Predict likelihood parameters for distance function objects

Usage

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

Arguments

`object`	An estimated dfunc object. See `dfuncEstim`.
`newdata`	A data frame containing new values of the covariates at which predictions are to be computed. If `newdata` is NULL, predictions are made at values of the observed covariates and results in one prediction (either parameters or distance function, see parameter `type`) for every observed distance. If `newdata` is not NULL and the model does not contains covariates, this routine returns one prediction (either parameters or distance function) for each row in `newdata`, but columns and values in `newdata` are ignored.
`type`	The type of predictions desired. • **If** `type` **= "parameters"**: Return predicted parameters of the likelihood function, one value for each observation (row) in `newdata`. If `newdata` is NULL, return one predicted parameter value for every detection in `object$detections`. • **If** `type` **is not "parameters"**: Return scaled distance functions. Distance functions are evaluated at the distances specified in `distances`. The number of distance functions returned depends on `newdata` and whether `object` contains covariates: – If `object` does NOT contain covariates, the distance function does not vary (by covariate) and only one distance function will be returned, even if `newdata` is specified. – If `object` contains covariates, one distance function will be returned for each observation (row) in `newdata`. If `newdata` is NULL, one distance function will be returned for every detection in `object$detections`. If `object` is a smoothed distance function, it does not have parameters and this routine will always return a scaled distance function. That is, `type` = "parameters" when `object` is smoothed does not make sense and the smoothed distance function estimate will be returned.

distances	A vector of distances 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, this routine uses a sequence of 200 evenly spaced distances between object$w.lo and object$w.hi, inclusive
...	Included for compatibility with generic predict methods.

Value

A matrix containing one of two types of predictions:

- **If** type **is "parameters"**, the returned matrix contains predicted likelihood parameters. The extent of the first dimension (rows) in the returned matrix is equal to either the number of detection distances in object$detections 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. Without expansion terms, the number of columns in the returned matrix is either 1 or 2 depending on the likelihood (e.g., halfnorm has one parameter, hazrate has two). See the help for each likelihoods to interpret the returned parameter values.
- **If** type **is not "parameters"**, the returned matrix contains scaled distance functions. The extent of the first dimension (rows) is either the number of distances specified in distance or 200 if distances is not specified. The extent of the second dimension (columns) is:
  - 1: if object does NOT contain covariates.
  - the number of detections: if object contains covariates and newdata is NULL.
  - the number of rows in newdata: if object contains covariates and newdata is specified.

  All distance functions in columns of the return are scaled to object$g.x.scale at object$x.scl.

  When type is not "parameters", the returned matrix has additional attributes containing the distances at which the functions are scaled and ESW's. attr(return, "x0") is the vector of distances at which each distance function in <return> is scaled. i.e., the vector of x.scl. attr(return, "scaler") is a vector scaling factors corresponding to each distance function in return. i.e., the vector of 1/f(x.scl) where f() is the unscaled distance function. If object contains line transects, attr(return, "scaler") is a vector of ESW corresponding to each distance function.

See Also

halfnorm.like, negexp.like, uniform.like, hazrate.like, Gamma.like

Examples

```
data(sparrowDetectionData)
data(sparrowSiteData)
# No covariates
dfuncObs <- dfuncEstim(formula = dist ~ 1
                     , detectionData = sparrowDetectionData
                     , w.hi = units::as_units(100, "m"))
predict(dfuncObs)
# values in newdata ignored because no covariates
predict(dfuncObs, newdata = data.frame(x = 1:5))
```

```
predict(dfuncObs, type = "dfunc") # one function

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

# Covariates
dfuncObs <- dfuncEstim(formula = dist ~ observer
                     , detectionData = sparrowDetectionData
                     , siteData = sparrowSiteData
                     , w.hi = units::as_units(100, "m"))
predict(dfuncObs)  # 356 X 1

Observers <- data.frame(observer = levels(sparrowSiteData$observer))
predict(dfuncObs, newdata = Observers) # 5 X 1

predict(dfuncObs, type = "dfunc") # 200 X 356
predict(dfuncObs, newdata = Observers, type = "dfunc") # 200 X 5
predict(dfuncObs, newdata = Observers, distances = d, type = "dfunc") # 3 X 5
```

`print.abund` *Print abundance estimates*

Description

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

Usage

```
## S3 method for class 'abund'
print(
  x,
  criterion = "AICc",
  maxBSFailPropForWarning = RdistanceControls()$maxBSFailPropForWarning,
  ...
)
```

Arguments

`x`	An object output by `abundEstim`. This is a distance function object that also contains abundance estimates, and has class `c("abund", "dfunc")`.
`criterion`	A string specifying the criterion to print. Must be one of "AICc" (the default), "AIC", or "BIC". See `AIC.dfunc` for formulas.
`maxBSFailPropForWarning`	The proportion of bootstrap iterations that can fail without a warning. If the proportion of bootstrap iterations that did not converge exceeds this parameter, a warning about the validity of CI's is issued and a diagnostic message printed. Increasing this to a number greater than 1 will kill the warning, but ignoring

a large number of non-convergent bootstrap iterations may be a bad idea (i.e., validity of the CI is questionable).

... Included for compatibility to other print methods. Ignored here.

Details

The default print method for class 'dfunc' is called, then the abundance estimates contained in `obj` are printed.

Value

No value is returned.

See Also

`dfuncEstim`, `abundEstim`

Examples

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

# Fit half-normal detection function
dfunc <- dfuncEstim(formula=dist ~ 1 + offset(groupsize)
                  , detectionData=sparrowDetectionData)

# Estimate abundance given a detection function
# Note: a person should do more than R=20 bootstrap iterations
fit <- abundEstim(dfunc
                , detectionData = sparrowDetectionData
                , siteData = sparrowSiteData
                , area = units::set_units(4105, "km^2")
                , R=20
                , ci=0.95)

# Print results
print(fit)
```

`print.dfunc` *Print a distance function object*

Description

Print method for distance functions produced by `dfuncEstim`, which are of class `dfunc`.

Usage

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

Arguments

x	An estimated distance function resulting from a call to dfuncEstim.
criterion	A string specifying the 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

The call, coefficients of the distanced function, whether the estimation converged, the likelihood and expansion function, and other statistics are printed. At the bottom of the output, the following quantities are printed,

- 'Strip' : The left (w.lo) and right (w.hi) truncation values.
- 'Effective strip width or detection radius' : ESW or EDR as computed by effectiveDistance.
- '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 printed is controlled by options()$digits.

Value

The input value of obj is invisibly returned.

See Also

dfuncEstim, plot.dfunc, print.abund

Examples

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

# Fit half-normal detection function
dfunc <- dfuncEstim(formula=dist~1,
                    detectionData=sparrowDetectionData)

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

RdistanceControls *Control parameters for* Rdistance *optimization.*

Description

Returns a list of optimization controls used in Rdistance and provides a way to change them if needed.

Usage

```
RdistanceControls(
  optimizer = "nlminb",
  evalMax = 2000,
  maxIters = 1000,
  likeTol = 1e-08,
  coefTol = 1.5e-08,
  hessEps = 1e-08,
  requireUnits = TRUE,
  maxBSFailPropForWarning = 0.2,
  contrasts = NULL
)
```

Arguments

optimizer	A string specifying the optimizer to use. Results vary between optimizers, so switching algorithms sometimes makes a poorly behaved distance function converge. The 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 parameter boundaries may require use of "optim".
evalMax	The maximum number of objective function evaluations allowed.
maxIters	The maximum number of optimization iterations allowed.
likeTol	The maximum change in the likelihood (the objective) between iterations that is tolerated during optimization. If the likelihood changes by less than this amount, optimization stops and a solution is declared.
coefTol	The maximum change in the model coefficients between iterations that is tolerated during optimization. If the sum of squared coefficient differences changes by less than this amount between iterations, optimization stops and a solution is declared.
hessEps	A vector of parameter distances used during computation of numeric second derivatives. Should have length 1 or the number of parameters in the model. See function secondDeriv.
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 transect length and study area size. Assign units with statement like units(detectionDf$dist)

<- "m" or units(df$transectDf) <- "km". Measurement units do not need to be the same. All units are converted appropriately during internal computations. Rdistance recognizes units listed in units::valid_udunits().

maxBSFailPropForWarning
: The proportion of bootstrap iterations that can fail without a warning. If the proportion of bootstrap iterations that did not converge exceeds this parameter, a warning about the validity of CI's is issued in the print method for abundance objects.

contrasts
: A list, whose entries are values (numeric matrices, functions or character strings naming functions) to be used as replacement values for the default contrasts function and whose names are the names of columns of data containing factors.

There are several ways to change the contrasts used for factors in Rdistance because all methods used in linear models (lm) work. To summarize contrast methods in R, if this parameter is NULL, Rdistance uses the global contrasts specified in options(). To change the global contrasts, use a statement like options(contrasts = c(unordered = "contr.SAS",ordered = "contr.poly")). One can also set contrasts for a factor using contrasts(a) (e.g., contrasts(a) <- "contr.sum") Lastly, one can set this parameter to a list that explicitely states the non-global contrasts to use for which factors in the Rdistance model. For example, list(a = "contr.helmert") will use Helmert contrasts for a and the global contrast option for all other factors. The built-in R contrast functions are "contr.treatment", "contr.helmert", "contr.SAS", "contr.sum", and "contr.poly".

Value

A list containing named components for each of the controls. This list has the same components as this function has input parameters.

Examples

```
# increase number of iterations
RdistanceControls(maxIters=2000)

# change optimizer and decrease tolerance
RdistanceControls(optimizer="optim", likeTol=1e-6)
```

---

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

Examples

```
set.seed(883839)
  x <- rnorm(1000) * 100
  x <- x[ 0 < x & x < 100 ]
  simp.expn <- simple.expansion(x, 4)
```

---

smu.like *Smoothed likelihood function for distance analyses*

Description

Computes the likelihood of sighting distances given a kernel smooth of the histogram.

Usage

```
smu.like(
  a,
  dist,
  covars = NULL,
  w.lo = 0,
  w.hi,
  scale = TRUE,
  series = NULL,
  expansions = 0,
  pointSurvey = FALSE
)
```

Arguments

a	A data frame containing the smooth. This data frame must contain at least an $x and $y components. These components are generally the output of function density.
dist	A numeric vector containing the observed distances.
covars	Not used in smoothed distance functions. Included for compatibility with other distance likelihoods in Rdistance.
w.lo	Scalar value of the lowest observable distance. This is the *left truncation* of sighting distances in dist. Same units as dist. Values less than w.lo are allowed in dist, but are ignored and their contribution to the likelihood is set to NA in the output.

w.hi	Scalar value of the largest observable distance. This is the *right truncation* of sighting distances in dist. Same units as dist. Values greater than w.hi are allowed in dist, but are ignored and their contribution to the likelihood is set to NA in the output.
scale	Logical scalar indicating whether or not to scale the likelihood so it integrates to 1. This parameter is used to stop recursion in other functions. If scale equals TRUE, a numerical integration routine (integration.constant) is called, which in turn calls this likelihood function again with scale = FALSE. Thus, this routine knows when its values are being used to compute the likelihood and when its value is being used to compute the constant of integration. All user defined likelihoods must have and use this parameter.
series	Not used in smoothed distance functions. Included for compatibility with other distance likelihoods in Rdistance.
expansions	Not used in smoothed distance functions. Included for compatibility with other distance likelihoods in Rdistance.
pointSurvey	Boolean. TRUE if distances in dist are radial from point transects, FALSE if distances are perpendicular off-transect distances.

Details

The approx function is used to evaluate the smooth function at all sighting distances.

Distances outside the range w.lo to w.hi are set to NA and hence not included.

Value

A numeric vector the same length and order as dist containing the likelihood contribution (height of the smoothed function) for all distances in dist. Assuming L is the vector returned by this function, the negative log likelihood of the sighting distances is -sum(log(L), na.rm=T). Note that the returned likelihood value for distances less than w.lo or greater than w.hi is NA, hence na.rm=TRUE in the sum. If scale = TRUE, the area under the smoothed curve between w.lo and w.hi is 1.0. If scale = FALSE, the integral of the smoothed curve is something else.

See Also

dfuncSmu, hazrate.like, uniform.like, negexp.like, halfnorm.like

Examples

```
set.seed(238642)
d <- units::set_units(abs(rnorm(100)), "in")
dfunc <- dfuncSmu(d~1)

L <- smu.like(a=dfunc$parameters,
       dist=dfunc$detections$dist,
       w.lo=dfunc$w.lo,
       w.hi=dfunc$w.hi,
       scale=TRUE)
-sum(log(L), na.rm=TRUE)  # the negative log likelihood
```

`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. See the package vignettes for tutorials of the basic analysis.

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`

`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. See the package vignettes for tutorials of the basic analysis.

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`

`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. See package vignettes for tutorials of the basic analysis.

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`

`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. See package vignettes for tutorials of the basic analysis.

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`

uniform.like *uniform.like - Uniform distance likelihood*

Description

Compute uniform-like distribution for distance functions. This function was present in Rdistance version < 2.2.0. It has been replaced by the more appropriately named logistic.like.

Usage

```
uniform.like(
  a,
  dist,
  covars = NULL,
  w.lo = 0,
  w.hi = max(dist),
  series = "cosine",
  expansions = 0,
  scale = TRUE,
  pointSurvey = FALSE
)
```

Arguments

a	A vector of likelihood parameter values. Length and meaning depend on whether covariates and expansions are present as follows: • If no covariates and no expansions: a = [a, b] (see Details) • If no covariates and *k* expansions: a = [a, b, e1, ..., e*k*] • If *p* covariates and no expansions: a = [a, b, b1, ..., b*p*] • If *p* covariates and *k* expansions: a = [a, b, b1, ..., b*p*, e1, ..., e*k*]
dist	A numeric vector containing observed distances with measurement units.
covars	Data frame containing values of covariates at each observation in dist.
w.lo	Scalar value of the lowest observable distance, with measurement units. This is the *left truncation* sighting distance. Values less than w.lo are allowed in dist, but are ignored and their likelihood value is set to NA in the output.
w.hi	Scalar value of the largest observable distance, with measurement units. This is the *right truncation* sighting distance. Values greater than w.hi are allowed in dist, but are ignored and their likelihood value is set to NA in the output.
series	A string specifying the type of expansion to use. Currently, valid values are 'simple', 'hermite', and 'cosine'; but, see dfuncEstim about defining other series.
expansions	A scalar specifying the number of terms in series. Depending on the series, this could be 0 through 5. The default of 0 equates to no expansion terms of any type.

scale	Logical scalar indicating whether or not to scale the likelihood into a density function, i.e., so that it integrates to 1. This parameter is used to stop recursion in other functions. If scale equals TRUE, a numerical integration routine (integration.constant) is called, which in turn calls this likelihood function again with scale = FALSE. Thus, this routine knows when its values are being used to compute the likelihood and when its values are being used to compute the constant of integration. All user defined likelihoods must have and use this parameter.
pointSurvey	Boolean. TRUE if dist is point transect data, FALSE if line transect data.

Value

A numeric vector the same length and order as dist containing the likelihood contribution for corresponding distances in dist. Assuming L is the returned vector, the log likelihood of all data is -sum(log(L), na.rm=T). Note that the returned likelihood value for distances less than w.lo or greater than w.hi is NA, and thus it is essential to use na.rm=TRUE in the sum. If scale = TRUE, the integral of the likelihood from w.lo to w.hi is 1.0. If scale = FALSE, the integral of the likelihood is arbitrary.

Expansion Terms

If expansions = k (k > 0), the expansion function specified by 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, the likelihood contribution for the i^{th} distance is,

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

---

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

Description

DEPRECATED.: Starting values and limits for parameters of the uniform distance function.

Usage

```
uniform.start.limits(dist, covars, expansions, w.lo, w.hi)
```

Arguments

dist	A numeric vector containing observed distances with measurement units.
covars	Data frame containing values of covariates at each observation in dist.
expansions	A scalar specifying the number of terms in series. Depending on the series, this could be 0 through 5. The default of 0 equates to no expansion terms of any type.

`w.lo` Scalar value of the lowest observable distance, with measurement units. This is the *left truncation* sighting distance. Values less than `w.lo` are allowed in `dist`, but are ignored and their likelihood value is set to `NA` in the output.

`w.hi` Scalar value of the largest observable distance, with measurement units. This is the *right truncation* sighting distance. Values greater than `w.hi` are allowed in `dist`, but are ignored and their likelihood value is set to `NA` in the output.

Details

This function is usually called within `F.start.limits`.

Value

A list containing the following components:

- `start` : a vector of starting values
- `lowlimit` : a vector of lower limits (can be -Inf)
- `highlimit` : a vector of upper limits (can be Inf)
- `nms` : a vector containing names of the parameters

Index