Package: dwp (via r-universe)

August 23, 2024

Type Package

Title Density-Weighted Proportion

Version 1.1

Date 2023-06-30

Description Fit a Poisson regression to carcass distance data and integrate over the searched area at a wind farm to estimate the fraction of carcasses falling in the searched area and format the output for use as the dwp parameter in the 'GenEst' or 'eoa' package for estimating bird and bat mortality, following Dalthorp, et al. (2022) <arXiv:2201.10064>.

Depends R (>= 3.6.0)

License CC0

LazyData true

Encoding UTF-8

Imports boot, expint, GenEst, gtools, invgamma, magrittr, MASS, matrixStats, methods, mvtnorm, numDeriv, plotrix, pracma, sf, statmod, VGAM

Suggests gpclib

RoxygenNote 7.2.3

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

RemoteUrl https://github.com/ddalthorp/dwp

RemoteRef HEAD

RemoteSha c3ef6e14d043cfceff8e9c61428b5e52c07b2de0

Contents

`Acins` *Calculate Area of Intersection inside Circle and Square with Common Center*

Description

Calculate Area of Intersection inside Circle and Square with Common Center

Usage

```
Acins(r, s)
```

Arguments

`r`	radius of the circle (vector or scalar)
`s`	half-width of the square (scalar)

Value

vector of intersections of interiors of circles with squaure

Examples

```
 # calculate area in annulus intersecting square
s <- 10 # radius or half-width of square
r <- c(11, 12) # inner and outer radii of circle
diff(Acins(r, s)) # intersection of square and annulus
# figure to illustrate the calculated area:
theta <- seq(0, 2 * pi, length = 1500)
plot(0, xlim = max(r) * c(-1, 1), ylim = max(r) * c(-1, 1),
  xlab = "x", ylab = "y", asp = 1, bty = "n", type = "n")
xi <- r[1] * cos(theta)
yi <- r[1] * sin(theta)
xo <- r[2] * cos(theta)
yo <- r[2] * sin(theta)
i1 <- which(abs(xi) <= s & abs(yi) <= s)
```

```
i2 <- which(abs(xo) <= s & abs(yo) <= s)
i2 <- sort(i2, decreasing = TRUE)
xi <- xi[i1]
yi <- yi[i1]
xo <- xo[i2]
yo <- yo[i2]
polygon(col = 8, border = NA,
  x = c(xi[xi >= 0 & yi >= 0], xo[xo >= 0 & yo >= 0]),
  y = c(yi[xi >= 0 & yi >= 0], yo[xo >= 0 & yo >= 0]))
polygon(col = 8, border = NA,
  x = c(xi[xi <= 0 & yi >= 0], xo[xo <= 0 & yo >= 0]),
  y = c(yi[xi <= 0 & yi >= 0], yo[xo <= 0 & yo >= 0]))
polygon(col = 8, border = NA,
  x = c(xi[xi <= 0 & yi <= 0], xo[xo <= 0 & yo <= 0]),
 y = c(yi[xi <= 0 & yi <= 0], yo[xo <= 0 & yo <= 0]))
polygon(col = 8, border = NA,
 x = c(xi[xi >= 0 & yi <= 0], xo[xo >= 0 & yo <= 0]),
 y = c(yi[xi >= 0 & yi <= 0], yo[xo >= 0 & yo <= 0]))
lines(r[1] * cos(theta), r[1]* sin(theta))
lines(r[2]* cos(theta), r[2] * sin(theta))
rect(-s, -s, s, s)
 # calculate areas in series of 1 m annuli extending to corner of square
 s <- 10.5 # radius of square (center to side)
 diff(Acins(r = 0:ceiling(sqrt(2) * s), s))
```

---

addCarcass *Add Carcasses to a Site Layout*

Description

After the site layout is analyzed and structured by rings for analysis, carcass data may still need to be added to the site data. addCarcass grabs carcass location data from a shape file or data frame and formats it into ring data, with carcass tallies in every 1m ring from the turbine to the maximum search distance away from any turbine.

Usage

```
addCarcass(x, ...)

## S3 method for class 'shapeCarcass'
addCarcass(
  x,
  data_ring,
  plotLayout = NULL,
  ncarcReset = TRUE,
  ccCol = NULL,
  ...
)
```

```
## S3 method for class 'data.frame'
addCarcass(
  x,
  data_ring,
  ccCol = NULL,
  ncarcReset = TRUE,
  unitCol = "turbine",
  rCol = "r",
  ...
)
```

Arguments

x	carcass data to insert into data_ring
...	ignored
data_ring	ring data for receiving carcass data from x
plotLayout	(optional) shapeLayout object to facilitate proper insertion of carcass data into ring structure.
ncarcReset	boolean to direct the function to set the carcass counts in all the rings to 0 before adding the new carcasses (default) or to add the new carcasses to the old totals (ncarcReset = FALSE).
ccCol	name of carcass class column (optional). Typically, the "carcass class" would be for carcass characteristics that would be expected to affect distances that carcasses would fall from the turbine. For example, distances would not be expected to be the same for large and small carcasses, and bats may have significantly different distance distributions than small birds. The ccCol could also be used for subsetting by any covariate that would be expected to interact with carcass distance distributions, like season (if winds vary by season) or turbine type (if the site has a diverse mix of turbines of different sizes or types). Additionally, ccCol can be used to subset the data by area (for example, NW, NE, SW, SE; or hilltop, river bank) or any other discrete covariate that the user may be interested in.
unitCol	name of unit column
rCol	name of column with carcass distances

Value

an object of class rings with a tally of the number of of carcasses discovered in each concentric 1m ring from the turbine to the most distant point searched.

Examples

```
data(layout_simple)
data(carcass_simple)
sitedata <- initLayout(layout_simple)
ringdata <- prepRing(sitedata)
ringsWithCarcasses <- addCarcass(carcass_simple, data_ring = ringdata)
```

aic — *Calculate Akaike Information Criterion (AICc) for Distance Distributions*

Description

functions for calculating AICc for carcass dispersion models.

Usage

```
aic(x, ...)

## S3 method for class 'ddArray'
aic(x, extent = "full", ...)

## S3 method for class 'ddArraycc'
aic(x, extent = "full", ...)

## S3 method for class 'dd'
aic(x, ...)
```

Arguments

x	list of models (ddArray) or single model (dd) to calculate AICs for
...	ignored
extent	Include only the extensible models (extent = "full") or all models (extent = "win"), whether or not they can be extended beyond the search radius.

Value

Data frame with AICc and deltaAICc for all models in x

alt_names — *Names of the Named Distributions*

Description

Names of the Named Distributions

Usage

```
alt_names
```

Format

An object of class character of length 10.

carcass_polygon — *Example Carcass Distances to Accompany the Polygon Layout Data Set*

Description

Example Carcass Distances to Accompany the Polygon Layout Data Set

Usage

```
carcass_polygon
```

Format

A data frame illustrating an import format for carcass distances. There are columns with turbine IDs (`turbine`) and carcass distances from turbine (`r`). Distances (`r`) are in meters from the nearest turbine. The data set is used in the "polygon" example in the User Guide.

carcass_simple — *Carcass Data to Accompany the Simple Geometry Data Format*

Description

Carcass Data to Accompany the Simple Geometry Data Format

Usage

```
carcass_simple
```

Format

An object of class `data.frame` with 55 rows and 2 columns.

carcass_simple0 — *Full, Simulated carcass_simple Data Set (Including Locations and Missed Carcasses)*

Description

Full, Simulated carcass_simple Data Set (Including Locations and Missed Carcasses)

Usage

```
carcass_simple0
```

Format

An object of class `data.frame` with 100 rows and 6 columns.

cof2parms — *Convert GLM Coefficients into Named Distribution Parameters*

Description

Convert GLM Coefficients into Named Distribution Parameters

Usage

```
cof2parms(x, ...)

## S3 method for class 'matrix'
cof2parms(x, distr, ...)

## S3 method for class 'numeric'
cof2parms(x, distr, ...)

## S3 method for class 'dd'
cof2parms(x, ...)

## S3 method for class 'matrix'
parms2cof(x, distr, ...)
```

Arguments

`x`	object (vector or matrix of parameters, dd, or glm) with named glm parameters (`"r"`, `"I(r^2)"`, `"I(r^3)"`, `"log(r)"`, or `"I(1/r)"`). NOTE: This function has minimal error-checking.
`...`	ignored
`distr`	name of the distribution

Value

matrix of parameters

`cofOK` *Check Whether GLM Coefficients Give Proper Distribution*

Description

In order for a fitted GLM to convert to a proper distance distribution, its integral from 0 to Inf must be finite. As a rule, when the leading coefficient is positive, the integral diverges as the upper bound of integration approaches infinity, and `cofOK` would return `FALSE`. Likewise, in some cases, the GLM coefficients yield an integral that diverges as the lower bound approaches 0, in which case `cofOK` returns `FALSE` as well. `cofOK0` and `cofOKInf` check the left and right tails of the candidate distribution, repectively, for convergence.

Usage

```
cofOK(cof, distr)

cofOK0(cof, distr)

cofOKInf(cof, distr)
```

Arguments

`cof`	vector or matrix of named glm parameters (with `"r"` as the distance variable)
`distr`	name of the distribution

Value

boolean vector (or scalar)

`cof_name` *Names of the GLM Coefficients for Each Distribution*

Description

Names of the GLM Coefficients for Each Distribution

Usage

```
cof_name
```

Format

A list with the names of the coefficients (vector of character strings) as they appear in the `$coefficients` value returned from `glm` for each model.

constraints *Constraints on GLM Coefficients for Extensibility to a Distribution*

Description

Constraints on GLM Coefficients for Extensibility to a Distribution

Usage

```
constraints
```

Format

A list of matrices giving the upper and lower bounds that each model's coefficients must meet for the model to be extensible to a distribution. The `parscale` column may be used in `optim` for fitting a truncated weighted likelihood model.

constraints_par *Constraints on Parameters to Assure Extensibility*

Description

Constraints on Parameters to Assure Extensibility

Usage

```
constraints_par
```

Format

An object of class `list` of length 17.

dd2ddSim — *Extract Parameters from a Distance Model (dd) and Format as ddSim Object*

Description

This is a utility function called internally by dwp functions to extract parameters from a fitted dd model and formats them for analysis and calculation.

Usage

```
dd2ddSim(dd)
```

Arguments

dd	dd object

Value

ddSim object with 1 row

ddCI — *Calculate CI for CDF, PDF, or quantile*

Description

Calculate a confidence interval for the CDF, PDF, or quantile of a carcass distance distribution.

Usage

```
ddCI(
  mod,
  x,
  type = "CDF",
  CL = 0.9,
  nsim = 1000,
  extent = "full",
  zrad = 200,
  na.tol = 0.1
)
```

Arguments

mod	a dd object
x	distance from turbine (scalar or vector) or probability (for quantile)
type	"CDF", "PDF", or "quantile"
CL	confidence level for the confidence interval(s)
nsim	number of simulation draws to base the estimate of CI on
extent	whether to calculate CI based on the full range of possible data and extrapolating beyond the search radius (extent = "full") or restricting the distribution to the area within the search radius (extent = "win").
zrad	an ad hoc radius to integrate to when the (uncommon) simulated parameter estimates do not result in an extensible distribution. In effect, This replaces NAs with 1s in CDFs and with 0s in PDFs.
na.tol	maximum fraction of invalid parameter sets to discard when constructing CIs; abort if mean(mod[, "extensible"]) > na.tol

Value

array (ddCI class) with columns for distance and the CI bounds

---

ddd *Calculate Probability Functions for Distance Distributions*

Description

Calculate the standard d/p/q/r family of R probability functions for distance distributions (dd) as well as the relative carcass density (rcd). Usage broadly parallels that of the d/p/q/r probability functions like dnorm, pnorm, qnorm, and rnorm.

Usage

```
ddd(x, model, parms = NULL, extent = "full", zrad = 200)

pdd(q, model, parms = NULL, extent = "full", zrad = 200, silent = FALSE)

qdd(p, model, parms = NULL, extent = "full", zrad = 200, subdiv = 1000)

rdd(n, model, parms = NULL, extent = "full", zrad = 200, subdiv = 1000)

rcd(x, model, parms = NULL, extent = "full", zrad = 200)
```

Arguments

x, q, p, n	numeric, $x \geq 0$
model	either a dd object or a ddSim object
parms	model parameters; required if model is specified as a character string rather than a dd or ddSim object (otherwise optional and ignored)
extent	for a full distribution extrapolated beyond the search radius to account for all carcasses, use extent = "full"; for a distribution restricted solely to carcasses falling within the search radius, use extent = "win".
zrad	the distance at which carcass density is assumed to be zero; to be used only in simulation reps in which simulated parameters do not yield extensible distributions, essentially returning 0 rather than NA for those pathological cases.
silent	If TRUE, then console messages are suppressed.
subdiv	if the number of values to calculate with rdd or qdd is >1, the function uses breaks the PDF into subdiv subdivisions and interpolates to solve the inverse. More subdivisions gives greater accuracy but is slower.

Details

The probability density function (PDF(x) = $f(x)$ = ddd(x, ...)) gives the probability that a carcass falls in a 1 meter ring centered at the turbine and with an outer radius of x meters. The cumulative distribution function [CDF(x) = $F(x)$ = pdd(x, ...)] gives the probability that a carcass falls within x meters from the turbine. For a given probability, p, the inverse CDF [qdd(p,...)] gives the p quantile of carcass distances. For example, qdd(0.5,...) gives the median carcass distance, and qdd(0.9, ...) gives the radius that 90% of the carcasses are expected to fall in. Random carcass distances can be generated using rdd.

The relative carcass density function(rcd) gives relative carcass densities at a point x meters from a turbine. In general, rcd is proportional to PDF(x)/x, normalized so that the surface of rotation of rcd(x) has total volume of 1. There are more stringent contstraints on the allowable parameters in the fitted (or simulated) glm's because the integral of PDF(x)/x must converge.

Distributions may be extrapolated beyond the search radius to account for all carcasses, including those that land beyond the search radius (extent = "full"), or may be restricted to carcasses falling within the searched area (extent = "win"). Typically, in estimating dwp for a fatality estimator like eoa or GenEst, the full distributions would be used.

The probability functions have a number of purposes. A few of the more commonly used are listed below.

PDF and CDF (ddd and pdd):

- to calculate the probability that carcass lands at a distance x meters from the turbine (or, more precisely, within 0.5 meters of x) or within x meters from the turbine, use a scalar value of x and a single model (dd or ddSim) with ddd or pdd, repspectively;
- to account for uncertainty in the probabilities at x, use ddd or pdd for with scalar x and a simulated set of parameters from the fitted model (ddSim object). This would be useful for calculating confidence intervals for the probabilities;
- to calculate probabilities for a range of x values according to a single model, use a vector x with a dd object or a ddSim object with one row. This would be useful for drawing graphs of PDFs or CDFs;

- to calculate simulated probabilites for a range of x values, use a vector x and a ddSim object of simulated parameter sets. This would be useful for drawing confidence regions around a fitted PDF or CDF.

Inverse CDF (qdd)**:**
- to calculate the distance that 100p% of the carcasses are expected to fall, use a scalar p in the interval (0, 1) and a single model (dd) or parameter set (ddSim with one row);
- to calculate account for the uncertainty in estimating the inverse CDF for a given p, use a scalar p and a ddSim object. This would be useful for calculating a confidence interval for, say, the median or the expected 90th percentile of carcass distances;
- to calculate the inverse CDF for a range of probabilities for a single model, use a vector p and a single model (dd or ddSim object with one row.

Random Carcasses Distances (rdd)**:**
- to generate n random carcass distances for a given (fixed) model, use a dd object or a ddSim object with a single row;
- to generate n random carcass distances for a model and account for the uncertainty in estimating the model, use a ddSim object with n rows, where n is also used as the n argument in the call to rdd.

Relative Carcass Density (per m^2):
- to calculate the relative carcass density at a number of distances, use a vector x. This would be useful in generating maps of carcass density at a site.

Value

vector or matrix of values; a vector is returned unless model is a ddSim object with more than one row and is to be calculated for more than one value (x, q, p), in which case an array with dimensions length(x) by nrow(model) is returned (where "x" is x, q, or p, depending on whether ddd, pdd, or qdd is called).

Examples

```
data(layout_simple)
data(carcass_simple)
sitedata <- initLayout(layout_simple)
ringdata <- prepRing(sitedata)
ringsWithCarcasses <- addCarcass(carcass_simple, data_ring = ringdata)
distanceModels <- ddFit(ringsWithCarcasses)
modelEvaluations <- modelFilter(distanceModels)
bestModel <- modelEvaluations$filtered
pdd(100, model = bestModel) # estimated fraction of carcasses within 100m
ddd(1:150, model = bestModel) # estimated PDF of the carcass distances
qdd(0.9, model = bestModel) # estimated 0.9 quantile of carcass distances
rdd(1000, model = bestModel) # 1000 random draws from estimated carcass distribution
```

---

ddFit *Fit Distance Distribution Model(s)*

Description

Fit generalized linear models (glm) for distance distribution models corresponding to standard forms [xep1, xep01 (gamma), xep2 (Rayleigh), xep02, xep12, xep012, xep123, xep0123 (normal-gamma with x = tau), lognormal, truncated normal, Maxwell Boltzmann, and constant] and supplentary forms [exponential, chi-squared, inverse gamma, and inverse Gaussian].

The glm is converted to a probability distribution by dividing by a normalizing constant, namely the integral of the glm evaluated from 0 to infinity. In some cases (most notably when the leading coefficient of the glm is positive so the fitted curve does not converge to zero as x increases), converted to a probability distribution. In these cases, the distribution parameters are given as NA, but the fitted model itself is saved.

Usage

```
ddFit(x, ...)

## S3 method for class 'data.frame'
ddFit(
  x,
  distr = "standard",
  scVar = NULL,
  rCol = "r",
  expoCol = "exposure",
  ncarcCol = "ncarc",
  silent = FALSE,
  ...
)

## S3 method for class 'rings'
ddFit(
  x,
  distr = "standard",
  scVar = NULL,
  rCol = "r",
  expoCol = "exposure",
  ncarcCol = "ncarc",
  silent = FALSE,
  ...
)

## S3 method for class 'list'
ddFit(
  x,
  distr = "standard",
  scVar = NULL,
  rCol = "r",
  expoCol = "exposure",
  ncarcCol = "ncarc",
  silent = FALSE,
```

```
  ...
)

## S3 method for class 'xyLayout'
ddFit(
  x,
  distr = "standard",
  scVar = NULL,
  notSearched = NULL,
  rCol = "r",
  ncarcCol = "ncarc",
  unitCol = "turbine",
  silent = FALSE,
  ...
)

## S3 method for class 'ringscc'
ddFit(
  x,
  distr = "standard",
  scVar = NULL,
  rCol = "r",
  expoCol = "exposure",
  ncarcCol = "ncarc",
  silent = FALSE,
  ...
)
```

Arguments

x	a search plot layout object to fit carcass distribution models to. The layout may be a data frame with columns for ring radii, exposure (or searched area in each ring), search class variable (optional), and number of carcasses in each ring;
...	ignored
distr	names (vector of character strings) of glm distribution templates to fit. Default is `distr = "standard"` to fit the standard models listed in the description above. Setting `distr = "all"` will fit both the standard models and the supplementary models. Also, any subset of the models may be fit by using, for example, `distr = c("xep01", "lognormal")` to fit only the `"xep01"` and `"lognormal"` models, or `distr = exclude(c("xep123", "constant"))` to fit all standard models except `"xep123"` and `"constant"`, or `distr = exclude("lognormal",mod_all)` to fit all the models except the lognormal.
scVar	Search class variable to include in the model (optional). `scVar` is ignored if `x` is not a `shapeLayout` or `xyLayout` object. If `x` is a `shapeLayout` object, `scVar` may be either `NULL` or the name of a single column with search class data. If `x` is an `xyLayout` object, `scVar` may be either NULL or a vector of names of search class variables to include in the models.

rCol	name of the distance column (which gives the outer radii of the rings). This will be correct by default for objects coming from prepRing and will rarely need to be explicitly specified.
expoCol	name of the column with the exposure, which is the area in the ring with outer radius rCol. This will be correct by default for objects coming from prepRing and will rarely need to be explicitly specified.
ncarcCol	name of the column with tallies of carcasses by ring. This will be correct by default for objects coming from prepRing and will rarely need to be explicitly specified.
silent	set silent = TRUE to suppress information printed to the console as the calculations proceed, which may be useful when running simulations.
notSearched	the name of the level (if any) in scVar that indicates an unsearched area
unitCol	name of the column with turbine IDs

Value

A list of fitted glm models as dd objects in a ddArray object if a vector of distributions is fit, or a single dd object if a single model is fit. The dd objects are lists that include the following elements:

glm the fitted model

$distr name of the distribution ("xep01", etc.)

$parms vector of distribution parameter estimates (or NA if the model based on the MLE is not extensible)

$varbeta the variance-covariance matrix of the glm parameter estimates. NOTE: This is identical to the covariance matrix from the glm, which can be extracted via summary(x)$cov.unscaled

$scVar name of the (optional) search class variable (or NULL)

$ncarc number of carcasses

$aicc the AICc value of the fit

$n number of rings

$k number of parameters

$srad search radius

When a dd object is printed, only a small subset of the elements are shown. To see a full list of the objects, use names(x). The elements can be extracted in the usual R way via $ or [[x]].

Examples

```
data(layout_simple)
data(carcass_simple)
sitedata <- initLayout(layout_simple) # initialize
ringdata <- prepRing(sitedata) # format site layout data for modeling
ringsWithCarcasses <- addCarcass(carcass_simple, data_ring = ringdata) # add carcasses to site
distanceModels <- ddFit(ringsWithCarcasses) # fit distance models
```

ddPrint *Print S3 Objects in* dwp *Package*

Description

dd, ddArray, and fmod objects are lists consisting of a great amount of data. Only a few of the elements are printed automatically. Other elements of object x can be viewed and extracted as with other lists in R, namely, by using the x$element or x[[element]] operator, where element is the name of one of the elements of x, all of which can be viewed via names(x).

Usage

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

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

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

Arguments

x	a ddArray or ddArray object
...	ignored

Value

no return value; output printed to the console

ddSim *Simulation of Dispersion Parameters*

Description

Simulation of Dispersion Parameters

Usage

```
ddSim(x, ...)

## S3 method for class 'dd'
ddSim(x, nsim = 1000, extent = "full", ...)
```

Arguments

x	object to simulate from
...	ignored
nsim	number of simulation draws
extent	simulate according to full distribution, including extrapolation beyond the search radius (extent = "full"); or restrict the distribution to the area within the search radius (extent = "win").

Value

array with simulated beta parameters from the glm model, and their conversion to distribution parameters

---

degOrder *An Ordering of the Models by Degree of the Polynomial*

Description

An Ordering of the Models by Degree of the Polynomial

Usage

```
degOrder
```

Format

An object of class character of length 17.

Distributions *Probability Distributions for Carcasses Versus Distance from Turbine*

Description

PDFs and CDFs that are required by ddd, pdd and qdd but are not included among the standard R distributions. Relying on custom code and included here are the Maxwell-Boltzmann (pmb and dmb), xep0 (Pareto), xep1, xepi0 (inverse gamma), xep2 (Rayleigh), xep02, xep12, xep012, xep123, and xep0123. Not included here are the distributions that can be calculated using standard probability functions from base R, namely the exponential, truncated normal, lognormal, gamma (xep01), and chisquared distributions and the inverse gaussian, which is calculated using statmod::dinvgauss and statmod::dinvgauss. The functions are designed for vector x or q and scalar parameters.

Usage

```
dmb(x, a)

pmb(q, a)

dxep1(x, b1)

pxep1(q, b1)

pxep02(q, b0, b2)

dxep02(x, b0, b2)

dxep12(x, b1, b2)

pxep12(x, b1, b2)

dxep123(x, b1, b2, b3, const = NULL)

pxep123(x, b1, b2, b3, const = NULL)

dxepi0(x, shape, scale)

pxepi0(x, shape, scale)

dxep0123(x, b0, b1, b2, b3, const = NULL)

pxep0123(x, b0, b1, b2, b3, const = NULL)

dxep012(x, b0, b1, b2, const = NULL)

pxep012(x, b0, b1, b2, const = NULL)

dxep2(x, s2)

pxep2(x, s2)

dxep0(x, a)

pxep0(x, a)
```

Arguments

x, q	vector of distances
a, b0, b1, b2, b3, shape, scale, s2	parameters used in the respective distributions.
const	(optional) scalar normalizing constant for distributions that are numerically integrated using integrate, namely. Providing a const is not necessary but will

improve the speed of calculation under certain conditions.

Details

An xep distribution is calculated by dividing its kernel (for the densities) or the integral of its kernel (for the cumulative distributions) by the normalizing constant, which is the integral of the kernel from 0 to Inf. The kernel of an xep distribution is defined as $xe^{P(x)}$, where $P(x)$ is a polynomial with terms defined by the suffix on xep. For example, the kernel of xep12 would be $xe^{b_1*x+b_2*x^2}$. A `0` in the suffix indicates a `log(X)` term and an `i` indicates a `1/x` term. The parameters of the xep distributions are some combination of b_0, b_1, b_2, b_3. The parameterizations of the inverse gamma (xepi0), Rayleigh (xep2), and Pareto (xep0) follow the standard conventions of `shape` and `scale` for the inverse gamma, `s2` = s^2 for the Rayleigh, and `a` = a for the Pareto (with a scale or location parameter of 1 and PDF = $a/x^(x+1)$ with support (1, Inf).

The Maxwell-Boltzmann is a one-parameter family with parameter `a` and PDF $f(a) = \sqrt{2/\pi}\frac{x^2e^{-x^2/(2a^2)}}{a^3}$. The kernel is $f(a) = x^2e^{-x^2}$, which has a simple closed-form integral that involves the error function (`pracma::erf`).

Value

vector of probability densities or cumulative probabilities

`distr_names` *Full List of Names of Distributions*

Description

Full List of Names of Distributions

Usage

```
distr_names
```

Format

An object of class `character` of length 17.

dwp *Density-Weighted Proportion*

Description

This package is designed to analyze carcass dispersion data and fit models of carcass density as function of distance from turbine.

Data sets

carcass_polygon
carcass_simple
layout_eagle
layout_polygon
layout_simple
layout_xy
xyr
sieve_default

Main Command-Line Functions

initLayout, prepRing, readCarcass, addCarcass import and format data

ddFit fit carcass distribution models

estpsi, estdwp estimate probability that carcass will lie in the searched area (psi) and the fraction of carcasses lying in the searched area ('dwp')

formatGenEst, exportGenEst format and export 'dwphat' objects for use with GenEst

aic, modelFilter, stats, ddCI statistics for fitted models

plot S3 function for ddArray, dd, fmod, polygonLayout, psiHat, dwpHat objects.

ddd, pdd, qdd, rdd, rcd probability functions for distance distributions

Potentially Useful Calculation and Editing Functions

ddSim, dd2ddSim functions for simulating dd models,

getncarc extract the number of carcasses per turbine from a data set; method for many types of objects,

cof2parms, cofOK, cofOK0, cofOKInf, constraints functions for manipulating and checking model coefficients

Acins calculate the area of the intersection of a circle and square sharing a common center

rmat, off functions for constructing functions out of distribution information

exclude simple function for excluding items from a superset

`estdwp` *Estimate DWP*

Description

Estimate the density-weighted proportion (DWP) of carcasses lying in the searched area at each turbine at a site. The calculation requires prior estimation of the expected proportion (`psi`) and the number of carcasses found (`ncarc`). NOTE: The carcass counts affect the uncertainty in the estimate of the fraction of carcasses in the searched area (DWP), and `ncarc` is required for accounting for uncertainty in estimates of DWP.

Usage

```
estdwp(x, ...)

## S3 method for class 'psiHat'
estdwp(x, ncarc, nboot = NULL, forGenEst = FALSE, silent = TRUE, ...)

## S3 method for class 'psiHatcc'
estdwp(x, ncarc, nboot = NULL, forGenEst = FALSE, silent = TRUE, ...)
```

Arguments

`x`	Either (1) `psiHat` object, which is an `nsim` by `nturbine` matrix that gives the estimated probability of that a given carcass will land in the searched area at each turbine, with turbine IDs as column names; or (2) a `psiHatcc` object, which is a list of `psiHat` objects, one for each carcass class.
`...`	ignored
`ncarc`	vector of total carcass count at each turbine represented in x.
`nboot`	number of parametric bootstrap iterations for estimating CIs
`forGenEst`	format the results for importing into GenEst (boolean)
`silent`	suppress messages from the fitting of a beta distribution in internal calculations that, if successful, increase the speed of the calculations by 20-200x. The message would signal that this acceleration cannot be applied.

Value

list

estpsi *Estimate Probability Carcass lands in Searched Area*

Description

Estimated probability that carcass lands in searched area. This is an intermediate step in estimating dwp but is also interesting in its own right. The estimation involves integrating the modeled carcass distribution (`model`) over the search plots at the turbines. Data for the search plots is stored in the generic argument, `x`, which can take any of a number of different forms, as described in the `Arguments` section (below).

Usage

```
estpsi(x, ...)

## S3 method for class 'rings'
estpsi(x, model, extent = "full", nsim = 1000, zrad = 200, ...)

## S3 method for class 'ringscc'
estpsi(
  x,
  model,
  modnames = NULL,
  extent = "full",
  nsim = 1000,
  zrad = 200,
  ...
)

## S3 method for class 'xyLayout'
estpsi(x, model, extent = "full", nsim = 1000, zrad = 200, ...)

## S3 method for class 'rpA'
estpsi(x, model, extent = "full", nsim = 1000, zrad = 200, ...)

## S3 method for class 'rdat'
estpsi(x, model, extent = "full", nsim = 1000, zrad = 200, ...)

## S3 method for class 'data.frame'
estpsi(x, model, extent = "full", nsim = 1000, zrad = 200, ...)
```

Arguments

`x` data

`rings` a formatted site map created from raw data via function `prepRing` (or as a component of a list returned by `addCarcass`).

ringscc a list of rings objects, one for each carcass class; created from raw data via function prepRing (or as a component of a list returned by addCarcass).

xyLayout formatted site map data derived from (x, y) coordinates covering every square meter of searched areas at each turbine; derived from the function initLayout, when called with xy data.

rpA (intended as an internal function that would rarely be called directly by users) a list of data frames (one for each turbine) giving the fraction of area searched (pinc at each distance (r). rpA data are embedded in rings objects that are created from site "maps" via prepRing.

rdat (intended as an internal function that would rarely be called directly by users) list of data frames giving the area searched ("exposure"), in a 1 meter ring with outer radius "r" and the number of carcasses found "ncarc" in each ring, with search class scVar optional. There is also a summary data frame $rdat[["total"]] that sums the exposures and carcass counts for all turbines across the site. The $rdat[["total"]] is the data frame used in fitting the GLMs. rdat objects are components of the return value of prepRing

data.frame (intended as an internal function that would rarely be called directly by users) a data frame giving the fraction of area searched (pinc at each distance (r).

... ignored

model A fitted dd model or an array of estimated parameters (ddSim object); or, if x is a ringscc object, a list of dd models (one for each carcass class), or a ddArraycc accompanied by a vector of model names to use (one for each carcass class).

extent calculate dwp within searched radius only ("win") or for full complement of carcasses ("full"), including those that fall outside the search radius.

nsim number of parametric bootstrap iterations for accounting for uncertainty in the estimator. Default is nsim = 1000. Use nsim = 0 for the estimate of psi based on the MLE of the given model without accounting for uncertainty.

zrad radius

modnames if x is a ringscc object, a vector of names of model to use for each carcass class; otherwise, modnames is ignored.

Value

A psiHat object, which is either 1) an array giving the expected fraction of carcasses lying in the searched area at each turbine with nsim rows and one column for each turbine + one row for the total; or 2) a list of such arrays, one for each carcass class if x is a ringscc object. The uncertainty in the expected fractions is characterized by simulation and reflected in the variation in psi values within each column.

exclude *Remove Particular Names from a Longer List*

Description

Removes specific values (what) from a longer vector of values (from). By default, from = mod_standard, and the intent is to simplify the subsetting of ddArray objects created with the default standard models. For example, dmod2 <- dmod[exclude("lognormal")] would subset the list of models in mod_standard to exclude "lognomal". The default can be overridden by providing a specific vector for from (for example, dmod[exclude("lognormal", from = names(dmod)]).

Usage

```
exclude(what, from = mod_standard)
```

Arguments

what	vector of distribution names to exclude
from	vector of distribution names to be excluded from

Value

vector of names from "from" after excluding "what"

exportGenEst *Export Estimated Density-Weighted Proportion to File in Proper GenEst Format*

Description

GenEst imports DWP from files with comma-separated values (.csv), with a column giving turbine ID and a column of DWP values for each carcass class. Column lengths are equal to the number of turbines times the number of simulation reps (typically, nsim = 1000), giving nsim copies of DWP values for each turbine in a single column.

Usage

```
exportGenEst(dwp, file)
```

Arguments

dwp	a dwphat object
file	name of file to export the dwp estimates to

Details

NOTE: The `.csv` file uses the English convention (used in USA, UK, Mexico, China, India, Australia, Japan, Korea, and others) with the comma (,) and not the semi-colon (;) to separate values among different columns and uses the period (.) as the decimal mark. Although GenEst can seamlessly accommodate either format, users in countries where the comma or other character is used as a decimal mark may need to adjust their software settings or edit the data to be able to view it in a spreadsheet program (such as Excel).

Value

The function writes the formatted data to a `.csv` file and returns NULL.

`fmmax` *Find suitable mmax for clipping improper priors for M*

Description

Improper priors need to be clipped in order to be usable. `fmmax` and `fmmax.ab` find values of m that are large enough that the probability of exceeding is less than 0.0001 (depends on g and X).

Usage

```
fmmax(x, g)

fmmax.ab(x, pBa, pBb)
```

Arguments

`x`	carcass count
`g`	overall carcass detection probability
`pBa, pBb`	parameters for beta distribution characterizing estimated g

Value

integer m such that $Pr(M >= m) < 0.0001$

formatGenEst *Format DWP Estimate for Use in GenEst*

Description

GenEst requires dwp data to be formatted as a data frame with columns for turbine ID and for estimated dwp for each carcass class. To incorporate uncertainty in the estimates, nsim simulated copies of the basic format are appended to the columns in the data set.

Usage

```
formatGenEst(dwphat)
```

Arguments

dwphat	a dwphat object

Value

an nsim*nturbine by nclass + 1 data frame, with columns for the turbine ID and for estimated dwp for each carcass class (e.g., large, medium, small, bat).

getncarc *Simple Function to Extract Carcass Counts*

Description

Carcass counts are easy to extract from any of the data structures, but it may be difficult to remember where to retrieve the data from for any particular structure. getncarc simplifies the task by having the same usage for all data types.

Usage

```
getncarc(x, ...)

## S3 method for class 'ringscc'
getncarc(x, ...)

## S3 method for class 'rings'
getncarc(x, ...)

## S3 method for class 'xyLayout'
getncarc(x, ...)

## S3 method for class 'ddArray'
getncarc(x, ...)
```

```
## S3 method for class 'ddArraycc'
getncarc(x, ...)

## S3 method for class 'dd'
getncarc(x, ...)
```

Arguments

x	a data structure with ncarc buried in it somewhere
...	ignored

Value

- scalar number of carcasses used in the fitted model (dd and ddArray objects
- vector of numbers of carcasses of each size used in the fitted models (ddArraycc objects)
- vector of carcass counts at each turbine and total at the site (xyLayout and rings objects
- list of vectors of carcass counts at each turbine for each carcass class

---

incGamma *Incomplete Gamma Function*

Description

Incomplete Gamma Function

Usage

```
incGamma(a, x, lower = FALSE)
```

Arguments

a	positive numeric vector
x	non-negative numeric vector
lower	boolean for calculating lower or upper incomplete gamma function

Details

The upper incomplete gamma function, following Wolfram Alpha, namely, incGamma(a, x) = $\int_x^\infty e^{-t} * t^{a-1} dt$, calculated using pgamma. Within the dwp package, incGamma is used in the calculation of the cumulative distribution function (CDF) of the xep02 distribution (pxep02). NOTE: The function pracma::incgam also calculates incomplete gamma with pracma::incgam(x, a) = incGamma(a, x), but pracma::incgam is not vectorized and not used here.

Value

scalar or vector of length = max(length(x), length(a)), with values of the shorter recycled to match the length of the longer a la pnorm etc.

initLayout *Create a Data Structure or Map for the Site Layout*

Description

Read plot layout data and perform premliminary error-checking and formatting. Search plot layout data can come in any of several different formats, including shape files for search area polygons and turbine locations, R polygons, (x, y) coordinates, or simple description of search plot type for each turbine (square, circular, road & pad). A vector of distances along with a search radius is also accommodated by dwp, but these can be directly processed in prepRing without preprocessing in initLayout.

Usage

```
initLayout(
  data_layout,
  dataType = "simple",
  unitCol = "turbine",
  file_turbine = NULL,
  radCol = "radius",
  shapeCol = "shape",
  padCol = "padrad",
  roadwidCol = "roadwidth",
  nRoadCol = "n_road",
  xCol = "x",
  yCol = "y",
  ncarcCol = "ncarc",
  scCol = NULL,
  notSearched = NULL,
  quiet = FALSE
)
```

Arguments

data_layout	Either the name of a shape file (polygons or multipolygons) that delineates areas searched at each turbine; a .csv file with R polygons, (x, y) coordinates, or simple descriptions of search parameters at each turbine; or a data frame with r polygons, (x, y) coordinates, or simple plot layout descriptions. See "Details" for details.
dataType	An identifier for the type of data to be imported: "shape", "polygon", "xy", or "simple". If data_layout is the name of a shape file, the dataType = "shape" identifier is optional.
unitCol	Column name for turbine IDs. If data_layout is the name of a shape file, then file_turbine must also be provided, giving turbine locations. The unitCol must be present in data_layout and in file_turbine (if provided). Turbine IDs in the unitCol must be syntactically valid R names (see Details below).

file_turbine	The name of a shape file (points) giving the turbine locations for turbines listed in the unitCol column in the data_layout if data_layout is a shape file. If dataType = "xy" and the grids in data_layout are all centered at (0, 0) with their turbines at the center, then file_turbine is not necessary and is ignored. Otherwise, if the grid coordinates are UTMs, file_turbine is either (1) a data frame with turbine names (in 'unitCol') and the location of turbines in 'x' and 'y', or (2) the name of a .csv file with turbine locations ('unitCol', 'x', and 'y'). file_turbine is not required (and is ignored) for other data types.
radCol	for dataType = "simple" layouts: the name of the column in data_layout that gives the search radius for each turbine
shapeCol	for dataType = "simple" layouts: the name of the column in data_layout that gives the plot shape for each turbine.
padCol	for dataType = "simple" layouts: the name of the column in data_layout that gives the radius of the turbine pad
roadwidCol	for dataType = "simple" layouts: the name of the column in data_layout that gives the width of the turbine access road(s)
nRoadCol	for dataType = "simple" layouts: the name of the column in data_layout that gives the number of turbine access roads at each turbine
xCol	for dataType = "xy" or dataType = "polygon" layouts: the name of the column in data_layout that gives x coordinates on the grid (for dataType = "xy") or x coordinates of search area polygon (for dataType = "polygon")
yCol	for dataType = "xy" or dataType = "polygon" layouts: the name of the column in data_layout that gives y coordinates on the grid (for dataType = "yy") or y coordinates of search area polygon (for dataType = "polygon")
ncarcCol	for dataType = "xy" layouts: the name of the column with carcass counts in each grid cell. The column is required but may be all zeros with carcasses added from a matrix of carcass locations later
scCol	for dataType = "xy" layouts: the name of column in data_layout with names of search classes. This is used for excluding unsearched areas from the grid data (x, y). It is used ONLY with dataType = "xy" and used to remove rows with x[, scCol] == notSearched, where x is the search grid data frame.
notSearched	for dataType = "xy" layouts: the name(s) of search class(es) in scCol that are not searched (optional). Ignored for data types other than xy.
quiet	boolean for controlling whether progress of calculations and other notes are printed to the console as the function runs

Details

All the layout types (except for vector, which is addressed elsewhere) can accommodate patterns of seached and not searched areas. If the searched areas are subdivided into different search classes with different detection probabilities, then search plot layout data must be input either from shape files with non-intersecting polgons delineating the search classes or from x-y grid data. If there is more than one search class variable (for example, ground cover and search schedule), then the covariates may be entered in separate columns if the layout files give grid coordinates or may be combined into one column in the shape files. For example, ground visibility may be easy or difficult

and search schedule may be 1-day or 14-day. These can be combined into a single column with values of, say, `easy1`, `easy14`, `difficult1`, and `difficult14`.

There must be a turbine ID column (`unitCol`) in each of the files. The individual turbine ID's must be syntactically valid R names, i.e., contain combinations of letters, numbers, dot (.), and underscores (_) only and not begin with a number or a dot followed by a number. Other characters are not allowed: no spaces, hyphens, parentheses, etc.

If shape files are to be imported, both shape files (search area polygons and turbine locations) must have their three standard, mandatory component files (.shp, .shx, .dbf) stored in the same folder. Only the name of the .shp should be entered in the function call. Other components are automatically searched for and processed if available.

Value

A list or data frame with components appropriate to the type of data imported. The data structure is returned as an S3 class object, which other functions in `dwp` can recognize and properly process. There is minimal processing on the data after importing, but the structures are lightly error-checked and formatted for more thorough processing, depending on data type and analysis objectives. Typically, the layout data will be later processed by a call to `prepRing` to create a characterization of the searched area at the site by "rings", with tallies of searched area, search classes, and fraction of area searched in concentric, 1 meter rings around each turbine. The format of the output depends on the format of the input. There are several possibilities, including, each of which is an S3 object with characteristics specific to the imported data:

`shapeLayout` List with elements:

- `$layout` = turbine search area configurations (polygons and multipolygons) from `data_layout` shape file as an `sf` object.
- `$layoutAdj` = polygons from `$layout` but recentered at (0, 0)
- `$turbines` = turbine data (as `sf` object)
- `$unitCol` = name of the column with turbine IDs (character string)
- `$tset` = turbine names (vector of character strings)
- `$tcenter` = locations of turbine centers (nturb by 2 array) with UTMs of turbine locations, extracted and simplified from `$turbines`. Column names are `X` and `Y`, measuring meters from a reference point. Row names are the names of the turbines.

`simpleLayout` Data frame with columns:

- `turbine` = turbine IDs (syntactically valid R names)
- `radius` = search radius. If `shape` = `"square"`, then radius is 1/2 the width of the square.
- `shape` = general descriptor of the shape of the search plot as `"square"`, `"circular"`, or `"RP"` (for roads and pads search).
- `padrad` = radius of the turbine pad (assumed circular)
- `roadwidth` = width of the access road(s)
- `n_road` = number of access roads

`polygonLayout` List of polygons, one for each turbine. The maximum search radius at any turbine is assigned as an attribute (`attr(, "rad")`).

`xyLayout` List with elements:

- xydat = data frame with columns for turbine names, x and y coordinates of 1m grid centers spanning the searched area, number of carcasses found in each grid cell, and optional covariates.
- tcenter = matrix giving turbine locations (x, y), with row names = turbine names.
- ncarc = vector giving the number of carcasses found at each turbine.
- unitCol = name of the column where turbine IDs are stored in xydat.
- tset = names of the searched turbines

Examples

```
data(layout_simple)
# converts properly formatted dataframe to 'simpleLayout' object
initLayout(layout_simple)

data(layout_xy)
initLayout(layout_xy, dataType = "xy")

data(layout_polygon)
initLayout(layout_polygon, dataType = "polygon", unitCol = "turbine")
```

---

layout_eagle *Example Bare Vector Format for Eagle Data*

Description

Example Bare Vector Format for Eagle Data

Usage

```
layout_eagle
```

Format

An object of class data.frame with 60 rows and 3 columns.

---

layout_polygon *Example Polygon Data for Site Layout*

Description

Example Polygon Data for Site Layout

Usage

```
layout_polygon
```

Format

A data frame illustrating the required raw data format for using standard R polygons to characterize a site layout. There must be three columns: one giving turbine IDs (`turbine`) and columns for the `x` and `y` coordinates that delineate the plot layouts. Turbine IDs must be syntactitally valid R names, that is, combinations of letters, numbers, underscores (_) and periods (.) and no spaces, hyphens, or other special characters. Names must not begin with a number, so `1, 2, 3, ...`, `3B1`, and `.72S` are NOT valid names. Coordinates shoud be in meters relative to a turbine at (0, 0).

---

layout_simple *Example Simple Geometry Data Format for Site Layout*

Description

Example Simple Geometry Data Format for Site Layout

Usage

```
layout_simple
```

Format

A data frame illustrating an import format for a simple description of a site layout by turbine. Each turbine (`turbine`) is classed according to the shape of its search plot, either `circular`, `square`, or `RP` (search on the roads and turbine pad only). For `circular` plots, all ground within `radius` meters of the turbine is searched. For `square` plots, the `radius` is half the width of the square along the x-axis, NOT the distance to the corner. For `RP` plots, the `radius` is the maximum distance searched on the roads. The geometry of the `RP` also includes a circular turbine pad with radius `padrad`, a road width of `roadwidth` meters, and the number of roads (`n_road`) searched out to `radius` meters from the turbine.

---

layout_xy *Example Data for Site Layout on an (x, y) Grid*

Description

Example Data for Site Layout on an (x, y) Grid

Usage

```
layout_xy
```

Format

A data frame illustrating the required raw data format for using a grid format to characterize a site layout. There are five columns: one giving turbine IDs (`turbine`), columns for the `x` and `y` coordinates on 1 m. grids that overlay the search plot, a column giving the carcass count in each grid cell, and a column giving the distance of each cell from the turbine. There is only one turbine, searched on road and pad out. Coordinates are in meters relative to the turbine at (0, 0).

`modelFilter` *Run Models through a Sieve to Filter out Dubious Fits*

Description

A set of fitted models (`ddArray`) is filtered according to a set of criteria that test for high AIC, high-influence points, and plausibility of the tail probabilities of each fitted distribution. `modelFilter` will either auto-select the best model according to a set of pre-defined, objective criteria or will will return all models that meet a set of user-defined, or default criteria. A table of how the models score according to each criterion is printed to the console.

Usage

```
modelFilter(dmod, sieve = "default", quiet = FALSE)
```

Arguments

`dmod`	a `ddArray` object
`sieve`	a list of criteria for ordering models
`quiet`	boolean to suppress (`quiet = TRUE`) or allow (`quiet = FALSE`) messages from `modelFilter`

Details

The criteria to test are entered in a list (`sieve`) with components:

1. `$rtail` = vector of probabilities that define a checkpoints on distributions to avoid situations where a model that may fit well within the range of data is nonetheless implausible because it predicts a significant or substantial probability of carcasses falling great distances from the nearest turbine. The default is to check whether or not a distribution predicts that less than 50% of carcasses fall within 80 meters, 90% within 120 meters, 95% within 150 meters, or 99% within 200 meters. Distributions that fall below any of these points (for example predicting only 42% within 80 meters or only 74% within 120 meters) fail the default `rtail` test. The format of the default for the test is `$rtail = c(p80 = 0.5, p120 = 0.90, p150 = 0.95, p200 = 0.99)`. Users may override the default by using, for example, `sieve = list(rtail = c(p80 = 0.8, p120 = 0.99, p150 = 0.99, p200 = 0.999))` in the argument list for a more stringent test or for a situation where turbines are small or winds are light. Alternatively, users may forego the test altogether by entering `sieve = list(rtail = FALSE)`. If specific probabilities are provided, they must be in a vector of length 4 with names `"p80"` etc. as in the examples above.

2. `$ltail` = vector of probabilities that define checkpoints on distributions to avoid situations where the search radius is short and a distribution that fits the limited data set well but crashes to zero just outside the search radius. The default is to check whether or not a distribution predicts that greater than 50% of carcasses fall with 20 meters or 90% within 50 meters. Distributions that pass above either of these checkpoints (for example predicting 61% of carcasses within 20 meters or 93% within 50 meters) are eliminated by the default `ltail` test. The format of the default for the test is `$ltail = c(p20 = 0.5, p50 = 0.90)`. Users may override the default by using, for example, `sieve = list(rtail = c(p20 = 0.6, p50 = 0.8))` in the argument list for a situation where it is known that carcasses beyond 50 meters are common.
3. `$aic` = a numeric scalar cutoff value for model's delta AICc scores. Models with AICc scores exceeding the minimum AICc among all the fitted models by `sieve$aic` or more fail the test. The default value is 10. Users may override the default by using, for example, `sieve = list(aic = 7)` in the argument list to use a delta AIC score of 7 as the cutoff or may forego the test altogether by setting `sieve = list(aic = FALSE)`
4. `$hin` = `TRUE` or `FALSE` to test for high influence points, the presence of which cast doubt on the reliability of the model. The function defines "high influence" as models with high leverage points, namely, points with $\frac{h}{1-h} > \frac{2p}{n-2p}$ (where h is leverage, p is the number of parameters in the model, and n is the search radius) with Cook's distance > `8/(n - 2*p)`. The criteria for high influence points were adapted from Brian Ripley's GLM diagnostics package `boot` (`glm.diag`). The test is perhaps most valuable in identifying distributions with high probability of carcasses landing well beyond what could reasonably be expected.

Several choices of pre-defined `sieves` are available (or, as described above, users may define their own criteria):

`sieve = "default"` The models are ordered by the following criteria:

1. extensibility
2. weight of right tail (discounting models that predict implausibly high proportions of carcasses beyond the search radius)
3. weight of the left tail (discounting models that predict implausibly high proportions of carcasses near the turbines)
4. AICc test (discounting models with delta AICc > 10)
5. high influence points (discounting models in which one or more of the data points exert a high influence on the fitted model, according to Ripley's GLM diagnostics package `boot` (`glm.diag`))
6. ranking by AICc

Precise definitions of the default sieve parameters are given in `sieve_default`.

`sieve = NULL` Returns a list of the extensible models without scoring them by other model selection criteria.

`sieve = "win"` Sorts models by high-influence points and AICc

`sieve = list(<custom>)` User provides a custom sieve, which may be a modification of the default sieve or de novo. To modify the default, use, for example, `sieve = list(hin = FALSE)` to disable the `hin` test but keep the other default tests, or `sieve = list(aic = 7)` to use 7 rather than 10 as the AIC cutoff, or `sieve = list(ltail = c(p20 = 0.3, p50 = 0.8))` to use a more stringent left tail test that requires CDF graphs to pass below the points (20, 0.3) and (50, 0.8). Custom `ltail` and `rtail` parameters must match the formats of the default tests, but

their probabilities may vary. To turn off the aic filter, use sieve = list(aic = Inf). To turn off the ltail filter, use sieve = list(ltail = c(p20 = 1, p50 = 1)). To turn off the rtail filter, use sieve = list(rtail = c(p80 = 0, p120 = 0, p150 = 0, p200 = 0)). These custom components may be mixed and matched as desired.

Value

An fmod object, which is an unordered list of extensible models if sieve = NULL; otherwise, a list of class fmod with following components:

$filtered the selected dd object or a ddArray list of models that passed the tests

$scores a matrix with all models tested (rownames = model names) and the results of each test (columns aic_test, rtail, ltail, hin, aic)

$sieve the test criteria, stored in a list with

- $aic_test = cutoff for AIC
- $hin = boolean to indicate whether high influence points were considered
- $rtail = numeric vector giving the probabilities that the right tail of the distribution must exceed at distances of 80, 120, 150, and 200 meters in order to pass
- $ltail = numeric vector giving the probabilities that the left tail of the distribution must NOT exceed at distances of 20 and 50 meters in order to pass

models a list (ddArray object) of all models tested

note notes on the tests

When a fmod object is printed, only a small subset of the elements are shown. To see a full list of the objects, use names(x), where x is the name of the fmod return value. The elements can be extracted in the usual R way via, for example, x$sieve or x[["sieve"]].

Examples

```
data(layout_simple)
data(carcass_simple)
sitedata <- initLayout(layout_simple)
ringdata <- prepRing(sitedata)
ringsWithCarcasses <- addCarcass(carcass_simple, data_ring = ringdata)
distanceModels <- ddFit(ringsWithCarcasses)
stats(distanceModels)
stats(distanceModels[["tnormal"]])
stats(distanceModels[["lognormal"]])
```

---

mod_all *Names of All the Available Models*

Description

Names of All the Available Models

Usage

```
mod_all
```

Format

An object of class `character` of length 17.

---

mod_color *Vector of Colors Used in Graphs of Fitted Models*

Description

Vector of Colors Used in Graphs of Fitted Models

Usage

```
mod_color
```

Format

An object of class `character` of length 17.

---

mod_lty *Vector of Line Types Used in Graphs of Fitted Models*

Description

Vector of Line Types Used in Graphs of Fitted Models

Usage

```
mod_lty
```

Format

An object of class `numeric` of length 17.

mod_offset *Vector of GLM Offsets for Available Models*

Description

Vector of GLM Offsets for Available Models

Usage

```
mod_offset
```

Format

An object of class `character` of length 17.

mod_standard *Vector of Names of Standard Models*

Description

Vector of Names of Standard Models

Usage

```
mod_standard
```

Format

An object of class `character` of length 12.

mod_xy *Vector of Names of Models Available for Grid Layout*

Description

Vector of Names of Models Available for Grid Layout

Usage

```
mod_xy
```

Format

An object of class `character` of length 8.

mpp2ddSim — *Convert Distribution Name + Parameters to* ddSim *Object*

Description

Utility function to format a distribution name and a vector of its parameter values to a ddSim object for use in the p/d/r/q family of functions

Usage

```
mpp2ddSim(distr, parms)
```

Arguments

distr	character string giving the name of one of the models fit by ddFit
parms	vector of parameters for the distr, or, alternatively, an array of parameter sets. Parameterization may follow either the GLM format or the distribution format. For example, the xep01 model (gamma distribution) has GLM parameters for log(r) and r, which are the coefficients of the polynomial in the xep01 format (namely, x * exp(b0*log(r) + b1*r)), or the gamma distribution parameters, shape and rate. The elements of parameter vector must be named. For example, parms = c(log(r) = -0.373, r = -0.0147) for the GLM format for an xep01 model or, equivalently, parms = c(shape = 1.63, rate = 0.0147) for the distribution format. If both formats are given, the GLM parameters are used and the distribution parameters ignored.

Value

a ddSim object with srad = NA

MpriorOK — *Check validity of format of custom prior for M*

Description

Check validity of format of custom prior for M

Usage

```
MpriorOK(prior)
```

Arguments

prior	a custom prior for M must be a matrix with columns for M and and associated probabalities P(M = m). The M column must begin at 0 and the probabilities must sum to 1.

Value

boolean. Is the prior formatted properly?

---

natural *Vector of Names of Models with Natural Offset*

Description

The natural offset is the area (m^2) searched at a given distance. Models that use the natural offset are referred to as "natural". Other models may use different offsets which alter the shape of the curve with distance in an a priori way that is unaffected by the data.

Usage

```
natural
```

Format

An object of class `logical` of length 17.

---

off *Utility Function for Constructing Offsets for GLMs*

Description

This is a simple utility function for calculating offsets when exposure is assumed to be 100 calculating fitted distributions (`PDF` and `CDF`) but cannot be used in the fitting of the distributions themselves because it does not account for incomplete search coverages at given distances.

Usage

```
off(r, distr)
```

Arguments

`r`	vector of distances
`distr`	name of the distribution to calculate the offset for

Value

A vector of offset values to use with distances `r` when fitting the `distr` model.

parm_name — *List of Names of the Distribution Parameters Associated with Respective Models*

Description

List of Names of the Distribution Parameters Associated with Respective Models

Usage

```
parm_name
```

Format

An object of class `list` of length 17.

parOK — *Check Parameter Value Validity the Distribution*

Description

Performs a ouick check on whether the parameters given in `parms` are valid for the `distr`.

Usage

```
parOK(parms, distr)
```

Arguments

`parms`	vector or matrix of named glm parameters (with `"r"` as the distance variable)
`distr`	name of the distribution

Value

vector (or scalar) of 0s, 1s, and 2s to indicate whether the parameters are non-extensible (i.e., flat-out bogus), valid for the distribution, or valid for the distribution and give finite point densities.

par_default *Default Graphics Parameters*

Description

Default Graphics Parameters

Usage

```
par_default
```

Format

An object of class `list` of length 65.

Plot *Plot dd and ddArray Objects*

Description

Plot CDF, PDF, or rcd (relative carcass density) for a single carcass dispersion glm model (`dd` object) or a list of models (`ddArray` object).

Usage

```
## S3 method for class 'ddArray'
plot(
  x,
  type = "CDF",
  extent = "full",
  distr = "all",
  xmax = NULL,
  resolution = 250,
  mod_highlight = NULL,
  ...
)

## S3 method for class 'dd'
plot(
  x,
  type = "CDF",
  extent = "full",
  xmax = NULL,
  resolution = 250,
  nsim = 1000,
```

```
  CL = 0.9,
  ...
)

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

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

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

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

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

Arguments

x	model(s) to plot
type	Type or representation of carcass dispersion to plot: "CDF", "PDF", or "rcd". The "CDF" gives the fraction of carcasses falling within r meters from a turbine and "PDF" is the associated probability density. The "rcd" gives the relative carcass density at a point r meters from a turbine and is PDF/(2 * pi * r).
extent	Plot dispersions as fraction of total carcasses ("full") or as fraction of carcasses within the searched area ("win").
distr	vector of names of distributions to plot or set = "all"
xmax	maximum distance to show in the graph; if xmax = NULL, the maximum distance is taken as the max distance in the data set to which the models were fit.
resolution	The number of line segments to break the curves into when plotting (i.e., x = seq(0, xmax, length.out = resolution)). Higher resolutions give smoother-looking curves.
mod_highlight	Character string giving the name of the model to highlight by plotting it last and with lwd = 2. If NULL, the curve associated with the lowest (best) AICc score is highlighted.
...	arguments that may be passed to plotting functions
nsim	Number of simulation reps to use for estimating confidence bounds for dd plot (ignored for ddArray objects)
CL	confidence level to show in a dd plot (ignored for ddArray objects)

Details

ddArray objects are plotted with lines in order of decreasing AICc, so that the "better" models are closer to the top and more prominent. The model with the lowest AICc ("best" model) is plotted last with a heavier line than the others.

For dd objects, the curve for the MLE of the parameters is plotted, along with a 100CL% confidence bounds determined for nsim simulation reps

The legend follows the ordering given by modelFilter with the default sieve or, if extent = "win" by (1) delta AICc < 10, (2) the absence of high-influence points, and (2) AICc. The best model according to the filter is listed first, with a heavier line than the others; the remaining distributions are listed in descending order, with the best models in the leftmost column.

Value

Plot displayed; no return value.

---

postM *Calculate posterior distribution of M and extract statistics (M* and CI)*

Description

Calculation of the posterior distribution of total mortality (M) given the carcass count, overall detection probability (g), and prior distribtion; calculation of summary statistics from the posterior distribution of M, including M* and credibility intervals.

Usage

```
postM(x, g, prior = "IbinRef", mmax = NA)

postM.ab(x, Ba, Bb, prior = "IbinRef", mmax = NULL)

calcMstar(pMgX, alpha)

MCI(pMgX, crlev = 0.95)
```

Arguments

x	carcass count
g	overall carcass detection probability
prior	prior distribution of M
mmax	cutoff for prior of M (large max requires large computing resources but does not help in the estimation)
Ba, Bb	parameters for beta distribution characterizing estimated g
pMgX	posterior distribution of M
crlev, alpha	credibility level $(1-\alpha)$ and its complement (α)

Details

The functions postM and postM.ab return the posterior distributions of $M|(X, g)$ and $M|(X, Ba, Bb)$, respectively, where Ba and Bb are beta distribution parameters for the estimated detection probability. postM and postM.ab include options to to specify a prior distribution for M and a limit for truncating the prior to disregard implausibly large values of M and make the calculations tractable in certain cases where they otherwise might not be. Use postM when g is fixed and known; otherwise, use postM.ab when uncertainty in g is characterized in a beta distribution with parameters Ba and Bb. The non-informative, integrated reference prior for binomial random variables is the default (prior = "IbinRef"). Other options include "binRef", "IbetabinRef", and "betabinRef", which are the non-integrated and integrated forms of the binomial and betabinomial reference priors (Berger et al., 2012). For $X > 2$, the integrated and non-integrated reference priors give virtually identical posteriors. However, the non-integrated priors assign infinite weight to $m = 0$ and return a posterior of $Pr(M = 0|X = 0, \hat{g}) = 1$, implying absolute certainty that the total number of fatalities was 0 if no carcasses were observed. In addition, a uniform prior may be specified by prior = "uniform". Alternatively, a custom prior may be given as a 2-dimensional array with columns for m and $Pr(M = m)$, respectively. The first column (m) must be sequential integers starting at $m = 0$. The second column gives the probabilities associated with m, which must be non-negative and sum to 1. The named priors ("IbinRef", "binRef", "IbetabinRef", and "betabinRef") are functions of m and defined on $m = 0, 1, 2, ...$ without upper bound. However, the posteriors can only be calculated for a finite number of m's up to a maximum of mmax, which is set by default to the smallest value of m such that $Pr(X \leq x|m, \hat{g}) < 0.0001$, where x is the observed carcass count, or, alternatively, mmax may be specified by the user.

Value

The functions postM and postM.ab return the posterior distributions of $M|(X, g)$ and $M|(X, Ba, Bb)$, respectively. The functions calcMstar and MCI return M^* value and credibility interval for the given posterior distribution, pMgX (which may be the return value of postM or postM.ab) and α value or credibility level.

---

prepmod *Internal Utility Function to Parse and Format Model for Calculating Psihat*

Description

Internal Utility Function to Parse and Format Model for Calculating Psihat

Usage

```
prepmod(model, nsim)
```

Arguments

model	dd or ddSim object
nsim	number of simulation reps. If nsim = 0, return dd2ddSim

Value

ddSim object of simulated distribution model parameters

---

prepRing *Format a Search Layout into Rings for Analysis*

Description

A function for creating a characterization of the search plot at each turbine by rings. The ground around each turbine is divided into 1 meter concentric rings out to the limit of the search plot. The amount of area searched in each ring and search class (if a search class column is present in the data) at each turbine is calculated, along with the fraction of area searched in each ring. In addition, sum totals of the area in each ring and the average fraction of the area searched in each ring across all turbines at the site are tallied as well. This is a convenient structure for the Poisson regressions that are used to estimate the carcass distributions with respect to distance from the turbines, the probabilities of carcasses landing in the searched areas, and the fraction of carcasses in the searched area.

Usage

```
prepRing(x, ...)

## S3 method for class 'shapeLayout'
prepRing(x, scVar = NULL, notSearched = NULL, silent = FALSE, ...)

## S3 method for class 'simpleLayout'
prepRing(x, ...)

## S3 method for class 'numeric'
prepRing(x, srad, ...)

## S3 method for class 'polygonLayout'
prepRing(x, ...)
```

Arguments

x	a search plot layout as imported and processed by initLayout into a shapeLayout, polygonLayout, or simpleLayout object, or a bare vector of carcass distances if search plots are all circular with the same radius and no unsearched area within the search radius.
...	ignored
scVar	name of the search class variable (optional), a column in the shape file for the search polygons. scVar is ignored if x is not a shapeLayout object.

notSearched	name of the search class(es) in scVar that represent unsearched areas. Applicable only if x is a shapeLayout object and scVar is provided. Polygons associated with scVar values in notSearched are not included in the rings characterization of the site. Also, turbines with no polygons that are not notSearched are not included in the rings.
silent	Processing shape files into rings may take several minutes. By default, prepRing prints periodic notice of the progress of the calculations for shape files. To suppress these notices, use silent = TRUE.
srad	search radius for data when x = bare vector of carcass observation distances.

Value

an object of class rings, which is a list with components

$rdat list of data frames giving the area searched ("exposure"), in a 1 meter ring with outer radius "r" and the number of carcasses found "ncarc" in each ring, with search class scVar optional. There is also a summary data frame $rdat[["total"]] that sums the exposures and carcass counts for all turbines across the site. The $rdat[["total"]] is the data frame used in fitting the GLMs.

$rpA list of data frames giving the proportion of area included in the searches ("pinc") in each ring ("r"). and the number of carcasses found "ncarc" in each ring, with search class scVar optional. There is also a summary data frame that sums the exposures and carcass counts for all turbines across the site. The $rpA data frames are used in estimating the probability of carcasses falling in the searched area at each turbine, which, in turn is used for calculating dwp

$srad the maximum search radius at any of the turbines

$ncarc vector of the number of carcasses at each turbine with names equal to the turbine names.

$scVar name of the search class variable(s) or NULL

$tcenter locations of turbine centers (nturb x 2 matrix) with UTMs of turbine locations. Column names are X and Y. Row names are the names of the turbines.

---

psi_extend *Simple Extension of a* dd *Model beyond the Search Radius*

Description

Extend a distance model beyond the search radius via multiplication by a fixed, assumed constant rather than the default normalization used for extensible models. psi_extend should not be used with psiHat objects that were calculated with extent = "full".

Usage

```
psi_extend(psi, fwin)
```

Arguments

psi	psiHat object
fwin	fraction of carcasses assumed to lie within the search radius. If psi includes psiHat for multiple carcass classes, fwin should be either a vector with one value for each carcass class so that length(psi) = length(fwin) or a scalar (which assumes all carcasses, regardless of carcass class, have the same probability of landing outside the search radius).

Value

psiHat object extended beyond the search radius

---

readCarcass *Import Carcass Observations Locations from Shape Files*

Description

Carcass coordiates (x, y) and turbine IDs are read using st_read and formatted for adding to rings data structures for analysis.

Usage

```
readCarcass(file_cod, unitCol = "turbine", quiet = FALSE)
```

Arguments

file_cod	name of the file with carcass observation data. Currently, the function requires a shape file, which gives the carcass locations on the same coordinate system that is used for the turbines. The geometry is a simple features points file, consisting of at least the three mandatory files standard components (.shp, .shx, .dbf) stored in the same directory. Only the name of the .shp is required (for example, file_cod = "carcasses.shp"). Other components are automatically searched for and processed if available.
unitCol	name of column with turbine IDs. Column name and turbine IDs must match those of the data_layout and file_turbine used in the call to initLayout.
quiet	boolean for directing the function to print calculation progress updates and messages to the console. This should be set to FALSE unless you know clearly why you want to turn off the messaging.

Value

a shapeCarcass object, which is a list with $carcasses, which is a sf representation of the shape file file_cod data; $unitCol, which is the name of the unit column; and $ncarc, which is a vector of carcass counts at the turbines listed in unitCol. The elements of the $ncarc are named by turbines at which they were found.

rmat *Simple Utility Function Used in Optimizing the GLM*

Description

A simple utility function that is used in fitting a GLM, creating a matrix of "x" values for use in the polynomial part of a xep-type model.

Usage

```
rmat(r, distr)
```

Arguments

r	vector of distances (>=0)
distr	name of the distribution

Value

array with `length(r)` rows and p columns, where p is the number of parameters in the glm (including the intercept). The first column is all 1s, and the remaining columns are functions of r, specifically, log(r), r, r^2, r^3, or 1/r, depending on what the distribution requires.

sieve_default *Test Criteria for Model Selection*

Description

Test Criteria for Model Selection

Usage

```
sieve_default
```

Format

A list containing the parameters used for test criteria in model selection an `ddArray` objects. The `sieve_default` values are used as a default in `modelFilter`. If desired, users may create their own tests, using `sieve_default` as a template. The same list elements must all be present and have the same structure as the defaults, namely:

`$aic` the cutoff for DeltaAIC scores; models with higher scores are removed from further consideration. Default is `$aic = 10`

`$hin` a boolean to indicate whether or not to use high leverage points as a criterion for model selection. Default is `$hin = TRUE`

`$rtail` a vector of probabilities that the fitted model must exceed at 80, 120, 150, and 200 meters. Default is `rtail = c(p80 = 0.50, p120 = 0.90, p150 = 0.95, p200 = 0.99)`. Custom test parameters must be a vector probabilities with "p80", "p120", "p150", and "p200" in the names.

`ltail` a vector of probabilities that a fitted model must not exceed at 20 and 50 meters. Default is `ltail = c(p20 = 0.50, p50 = 0.90)`. Custom test parameters must be a vector of probabilities with "p20" and "p50" in `names`.

`sieve_win` *Test Criteria for Model Selection within Search Area*

Description

Test Criteria for Model Selection within Search Area

Usage

```
sieve_win
```

Format

A list containing the parameters used for test criteria in model selection in `ddArray` objects. The `sieve_win` values are used when either `sieve = "win"` or `extent = "win"` in arg list of `modelFilter`. The sieve parameters are:

`$aic = 10` the cutoff for DeltaAIC scores; models with higher scores are removed from further consideration.

`$hin = T` a boolean to indicate whether or not to use high leverage points as a criterion for model selection.

`$rtail` Appropriate only for extrapolating beyond the search radius. Automatically disabled via`rtail = sieve_default$rtail * 0`.

`ltail` Appropriate only for extrapolating beyond the search radius. Automaticall disabled via `ltail = sieve_default$ltail * 0 + 1`

`stats` *Display a Tables of Summary Statistics for Distance Distributions*

Description

Calculate summary statistics for a single distance distribution (`dd` object), an array of distance distributions all fit to the same data set (`ddArray`), or a list of arrays of distance distributions fit for different carcass classes but the same site layout (`ddArraycc`).

Usage

```
stats(x, ...)

## S3 method for class 'dd'
stats(x, extent = "full", zrad = 200, ...)

## S3 method for class 'ddArray'
stats(x, extent = "full", zrad = 200, ...)

## S3 method for class 'ddArraycc'
stats(x, extent = "full", zrad = 200, ...)
```

Arguments

x	list of models (ddArray) or single model (dd) to calculate summary statistics for
...	ignored
extent	distributions within searched area ("win") or extended beyond ("full")
zrad	maximum distance that carcasses can lie (only used when glm parameters not extensible to Inf)

Value

list (or list of lists if x is ddArray) with $model giving the model parameters and $stats giving the median, and 75th, 90th, and 95th quantiles of carcass distances and the estimated probability a carcass falls within the search area according to each model

---

subset.shapeCarcass *Subset Data from an Imported and Formatted Shape File*

Description

Subset Data from an Imported and Formatted Shape File

Usage

```
## S3 method for class 'shapeCarcass'
subset(x, subset, select, ...)

## S3 method for class 'shapeLayout'
subset(x, subset, select, ...)
```

Arguments

x	object to be subsetted
subset	values to subset by. For example, to subset x to include only turbines "t1" and "t2", then subset = c("t1", "t2"). The name of the column with turbine names is given in select.
select	the name of the column with the values to subset by. For example, to subset x by turbines names "t1" and "t2" as found in the "turbine" column in the data, use select = "turbine" and subset = c("t1", "t2").
...	ignored

Value

object of the same class as x, subsetted to values of select equal to some element in subset.

---

xyr *Locations of All Carcasses in Grid Data*

Description

Locations of All Carcasses in Grid Data

Usage

```
xyr
```

Format

matrix with columns x, y, r for all 100 carcasses in the simulation to generate the carcass data for the xy grid that was searched on road and pad.

[.ddArray *Subset a Set of Fitted Dispersion Models (*ddArray*)*

Description

Subset a Set of Fitted Dispersion Models (ddArray)

Usage

```
## S3 method for class 'ddArray'
x[distr]
```

Arguments

x	object to subset
distr	vector of names or integer indices of distributions to extract from fitted models

Details

Subset the ddArray object as if it were a simple vector

Value

list of selected models with crucial statistics

[.ddSim *Subset Simulated Dispersion Parameters while Preserving Attributes*

Description

Subset Simulated Dispersion Parameters while Preserving Attributes

Usage

```
## S3 method for class 'ddSim'
x[i, j, ...]
```

Arguments

x	object to subset
i, j	row and column indices to subset
...	ignored

Details

Subset the ddSim object as if it were a simple matrix or array

Value

array with simulated beta parameters from the glm model, their conversion to distribution parameters. NOTE: subsetting to a column or a row returns a matrix rather than a vector. This simplifies the coding and makes it easier to maintain integrity of data structures, but behavior differs from what is done when subsetting standard R matrices and arrays to a single column or row. Also unlike with standard R arrays and matrices, the class structure and attributes are preserved upon subsetting.

Index