

# Package ‘windAC’

April 24, 2022

**Type** Package

**Title** Area Correction Methods

**Version** 1.2.8

**Date** 2022-04-07

**Author** Paul Rabie [aut],  
Daniel Riser-Espinoza [aut, cre],  
Trent McDonald [ctb]

**Depends** R (>= 3.6.0), sf

**Imports** mvtnorm

**Maintainer** Daniel Riser-Espinoza <despinoza@west-inc.com>

**Description** Post-construction fatality monitoring studies at wind facilities are based on data from searches for bird and bat carcasses in plots beneath turbines. Bird and bat carcasses can fall outside of the search plot. Bird and bat carcasses from wind turbines often fall outside of the searched area. To compensate, area correction (AC) estimations are calculated to estimate the percentage of fatalities that fall within the searched area versus those that fall outside of it. This package provides two likelihood based methods and one physics based method (Hull and Muir (2010) <[doi:10.1080/14486563.2010.9725253](https://doi.org/10.1080/14486563.2010.9725253)>, Huso and Dalthorp (2010) estimate the carcass fall distribution. There are also functions for calculating the proportion of area searched within one unit annuli, log logistic distribution functions, and truncated distribution functions.

**Encoding** UTF-8

**License** GNU General Public License

**RoxygenNote** 7.1.2

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2022-04-24 14:50:02 UTC

## R topics documented:

calcAC . . . . . 2

carcassDistance . . . . .	4
circleBoxInt . . . . .	5
estTWL . . . . .	6
estWD . . . . .	8
geometricRectanglePropSearchTable . . . . .	9
geometricRoadPadPropSearchTable . . . . .	11
getDistanceProbability . . . . .	12
getDistributionFunction . . . . .	14
getDistributionSummary . . . . .	14
getProportionAreaSearched . . . . .	15
getStartValue . . . . .	16
hallingstad . . . . .	17
hullMuirAreaCorrection . . . . .	18
hullMuirMaxDistance . . . . .	20
LogLogistic . . . . .	21
print.windAC . . . . .	23
proportionAreaSearched . . . . .	23
secondDerivative . . . . .	24
triangleProb . . . . .	25
truncatedDistribution . . . . .	26
turbineSpatial . . . . .	29
weightFun . . . . .	29
windAC . . . . .	31

## Index 33

---

calcAC	<i>Calculate the area correction value(s) with confidence intervals</i>
--------	---

---

### Description

Use a fitted carcass density distribution and data describing the search area to calculate area correction values and confidence intervals using a parametric bootstrap approach.

### Usage

```
calcAC(
  distribution,
  paramVec,
  varcovVec = NULL,
  proportionSearchDF,
  distanceCol,
  proportionCol,
  additionalCol = NULL,
  nBoot = NULL,
  truncBounds = NULL,
  ciLevel = 0.9,
  randomSeed = NULL,
```

```
    ...
  )
```

### Arguments

distribution	Character indicating the distribution, passed to <a href="#">getDistanceProbability</a> .
paramVec	Numeric vector for the parameters associated with distribution. Assumed to be in the same order as the function indicated by distribution.
varcovVec	Numeric vector for the variances and covariances for paramVec, default is NULL, see details.
proportionSearchDF	Data frame with at least two columns: distance of the outer edge of an annulus from turbine and proportion of area searched within each annulus.
distanceCol	Character indicating the column name for the distance from turbine
proportionCol	Character indicating the column name for the proportion of area searched.
additionalCol	Character vector, default is NULL, indicating additional columns of how the area correction value should be calculated, see details and examples.
nBoot	Integer, indicating the number of parametric bootstrap replicates to use. Default is NULL, and not confidence intervals are produced.
truncBounds	Numeric, indicating bounds for the area correction calculation, see details. Default is NULL, and the bounds are set to $c(0, Inf)$ .
ciLevel	Numeric, default is 0.9, desired confidence level for the bootstrap confidence interval.
randomSeed	Numeric value of random seed, default is NULL.
...	Additional arguments passed to <a href="#">getDistanceProbability</a> and <a href="#">rmvnorm</a> .

### Details

The function [getDistanceProbability](#) is used to calculate the probability (fraction of carcasses) in the intervals between distances in `proportionSearchDF`.

The `truncBounds` argument defaults to zero as a lower bound and infinity as the upper bound. If a single value is provided, it is assumed as the upper bound with zero as the lower bound. If two or more values are provided, the `max(truncBounds)` is the upper bound and `min(truncBounds)` is the lower bound.

If `varcovVec` is NULL, then parametric bootstrapping is impossible and a confidence interval is not estimated. The `varcovVec` should be in such an order that correctly fills the lower triangle including the diagonal. The first column is filled, then the second, and so on. This forms the variance-covariance matrix for the parameters.

If `nBoot` is greater than zero, a parametric bootstrap is done. Bootstrap parameters are generated using the [rmvnorm](#) function.

If the `additionalCol` argument is not NULL, separate area corrections are estimated for each unique value within the column.

### Value

windAC object

**See Also**

[weightedLikelihood](#) [weightedDistribution](#) [getDistanceProbability](#)

**Examples**

```
## proportion of area searched data
data(proportionAreaSearched)

## no parametric bootstrap
noBootstrap <- calcAC(distribution = 'gamma',
  paramVec = c(2.483323, 0.02495139),
  varcovVec = NULL,
  proportionSearchDF = proportionAreaSearched,
  distanceCol = 'distanceFromTurbine',
  proportionCol = 'proportionAreaSearched',
  additionalCol = 'plotType')

## with a parametric bootstrap
withBootstrap <- calcAC(distribution = 'gamma',
  paramVec = c(2.483323, 0.02495139),
  varcovVec = c(0.041189428, 0.0008825275, 2.118081e-05),
  proportionSearchDF = proportionAreaSearched,
  distanceCol = 'distanceFromTurbine',
  proportionCol = 'proportionAreaSearched',
  additionalCol = 'plotType',
  nBoot = 10)
```

---

carcassDistance

*Carcass distance example data set*

---

**Description**

Example data set of carcass distance found during searches under wind turbines.

**Usage**

```
data(carcassDistance)
```

**Format**

A data frame with 56 rows and 6 variables:

**season** The season in which that carcass was found.

**plotType** The plot type (either FULL or RP) that was being searched when the carcass was found.

**proportionTurbineType** The proportion of turbines that have that plot type. For plotType=FULL the value is 0.1 and 0.9 for plotType=RP.

**distanceFromTurbine** Meters, carcass distance from the turbine.

**compassBearingDegree** Degrees, the compass bearing from the turbine to the carcass location. Zero degrees is north, 90 degrees is east, etc.

**probabilityDetection** The summarized probability of detection from the bias trial information.

### Details

The fictitious wind farm has 100 turbines. For 90 turbines the search plot consists of the turbine pad and the road out to 100 meters (RP = road and pad). The remaining 10 turbines had full plots (FULL) that are circular with a radius of 100 meters. The wind farm was searched during two seasons: Spring and Fall.

The example carcasses are all the same size class (bats).

Bias trials are also done to account for searcher efficiency and carcass removal. These results are summarized as a probability of detection in the data set.

### References

<https://tethys.pnnl.gov/publications/us-fish-wildlife-service-land-based-wind-energy-guidelines>

---

circleBoxInt

*Integration of the intersection of a rectangle and a circle*

---

### Description

Calculates the area of the intersection between a rectangle and a circle.

### Usage

circleBoxInt(R, S, L, ...)

### Arguments

R	Numeric, circle radius.
S	Numeric, short side of the rectangle
L	Numeric, long side of the rectangle
...	Currently ignored.

### Details

The rectangle is defined with lower left corner being the origin and upper right corner at (L, S). The area returned is the intersection between the circle, centered at the origin, and the rectangle.

If  $R \leq S$  then  $(\pi R^2)/4$  is returned.

If  $R \geq \sqrt{S^2 + L^2}$  then  $L * S$  is returned.

If  $R \leq L$  then  $R^2 * \sin^{-1}(S/R)/2 + S * \sqrt{(R^2 - S^2)}/2$  This is the area of a circle in the first quadrant between the horizontal line  $y = S$

if  $R > L$  and  $R < \sqrt{S^2 + L^2}$  then

$$(R^2 * \sin^{-1}(S/R)/2 + S * \sqrt{(R^2 - S^2)}/2) - (R^2 * \sin^{-1}(B/R)/2 + S * \sqrt{(R^2 - B^2)}/2) + B * L$$

where  $B = \sqrt{R^2 - L^2}$ . In this case there is part of the circle to the right of the rectangle. First set of parenthesis is the area of the circle below S, the second set is the area below B. Subtracting the two gives the area between B and S. The rectangle defined by B and L needs to be added back in.

## Value

Numeric value

## Examples

```
radius <- 115
short <- 80
long <- 100
circleBoxInt(R=radius,S=short,L=long)

## not run
## the integral is the area inside the polygon

x <- seq(0,max(radius,long),length=100)
outlineY <- function(x,R,S,L){
  suppressWarnings(y <- sqrt(R^2-x^2))
  y[x>R] <- 0
  y[x>L] <- 0
  y[y>=S] <- S
  return(y)
}
y <- outlineY(x=x,R=radius,S=short,L=long)
plot(x,y,type='l',ylim=c(-10,short))
text(long,0,label='L',pos=1)
text(0,short,label='S',pos=1)
text(long,sqrt(radius^2-long^2),label='B',pos=4)
```

---

estTWL

*Truncated Weighted Likelihood Estimation*

---

## Description

Maximum likelihood estimation of a (possibly truncated) probability density function is completed with weights on the likelihood.

## Usage

```
estTWL(fatDist, fatW, distribution, plotBounds = NULL, ...)
```

```
weightedLikelihood(fatDist, fatW, distribution, plotBounds = NULL, ...)
```

**Arguments**

fatDist	Vector of fatality distances from the turbine.
fatW	Vector of weights, to weight the likelihood for estimation. This must be the same length as fatDist and is assumed to be in the same order as fatDist.
distribution	Character indicating the distribution for weightedLikelihood or vector for estTWL.
plotBounds	Vector of length 1 or 2. If the length is 2 (or greater) the max value is used as the upper truncation bound and the min value is used as the lower truncation bound. If the length is 1 this value is taken as the upper truncation bound and zero is set as the lower truncation bound. The default is NULL, in which case the bounds are zero and positive infinity.
...	Additional arguments passed to <code>optim</code> .

**Details**

The truncated likelihood for a single observation is

$$L^*(\theta|x_i) = \frac{f(x_i|\theta)}{\int_a^b f(y|\theta)dy}$$

Where  $x_i$  is fatDist,  $\theta$  is the vector of parameters to be estimated, a and b correspond to the plotBounds and  $f()$  is the distribution chosen.

The truncated weighted likelihood is

$$TWL(\theta|\underline{x}) = \prod_{i=1}^n L^*(\theta|x_i)^{w_i}$$

Where  $n = \text{length}(\text{fatDist})$  and  $w_i$  is fatW.

The truncated weighted likelihood is then estimated using standard maximum likelihood techniques.

See `estTWL` for examples.

**Value**

Data frame of the parameter estimates for the distribution with fit statistics.

**See Also**

[calcAC](#)

**Examples**

```
## load the data
data(carassDistance)
data(proportionAreaSearched)

## add proportion of area searched to each carcass
carcDist <- merge(carassDistance,proportionAreaSearched,
```

```

by=c('plotType','distanceFromTurbine'),all.x=TRUE)

## create the weight for each carcass
carcDist$w <- with(carcDist,1/(proportionAreaSearched*probabilityDetection))

twlOutput <- with(carcDist,estTWL(fatDist=distanceFromTurbine,fatW=w,plotBounds=c(0,100),
distribution=c('norm','weibull','gamma')))

```

---

estWD

*Weighted Distribution Estimation*


---

### Description

Maximum likelihood estimation of a weighted probability density function is completed. is done on a weighted distribution. The weighted distribution is a typical probability density distribution multiplied by a weight function. The weight function can be used to truncate the distribution by returning zero beyond some threshold value.

### Usage

```

estWD(fatDist, weightFun, distribution, ...)

weightedDistribution(fatDist, weightFun, distribution, ...)

```

### Arguments

fatDist	Vector of fatality distanes from the turbine.
weightFun	R function that is multiplied by the probability distribution, see details.
distribution	Character indicating the distribution for weightedDistribution or vector for estWD.
...	Additional arguments passed to weightFun or <a href="#">optim</a> .

### Details

The function estWD is a convient wrapper function to weightedDistribution, for fitting multiple distributions.

The weight function should return a (relative) probability of detection at every distance. Typically this is the proportion of area searched. The function [weightFun](#) is set up to take a table of proportion of area searched and return values in a function format.

Let  $h(x)$  be the weight function ([weightFun](#)),  $f(x|\theta)$  be a probability density function (specified by [distribution](#),  $x$  be the vector of carcass distances from the turbine ([fatDist](#)), and  $\theta$  be the parameter vector to be estimated. The weighted distribution is

$$f_d(x|\theta) = \frac{h(x)f(x|\theta)}{\int h(y)f(y|\theta)dy}$$



The likelihood that is maximized is

$$L_d(\theta|\underline{x}) = \prod_{i=1}^n \frac{h(x_i)f(x_i|\theta)}{\int h(y)f(y|\theta)dy}$$

### Value

Data frame of the parameter estimates with fit statistics.

### See Also

[calcAC](#)

[weightFun](#)

### Examples

```
## load the data
data(carcassDistance)
data(proportionAreaSearched)

#####
## fit for fall carcasses found on road and pad (RP)
distanceFallRP <- subset(carcassDistance,plotType=='RP'&season=='fall',
select=distanceFromTurbine,drop=TRUE)

fallRPfit <- estWD(fatDist=distanceFallRP,weightFun=weightFun,
distribution=c('norm','gamma','weibull'),propTable=proportionAreaSearched,type='RP',
typeCol='plotType',distanceCol='distanceFromTurbine',propCol='proportionAreaSearched',
maxDistance=100)

#####
## fit for fall carcasses found on full plots
distanceFallFP <- subset(carcassDistance,plotType=='FULL'&season=='fall',
select=distanceFromTurbine,drop=TRUE)

fallFPfit <- estWD(fatDist=distanceFallFP,weightFun=weightFun,
distribution=c('norm','gamma','weibull'),propTable=proportionAreaSearched,type='FULL',
typeCol='plotType',distanceCol='distanceFromTurbine',propCol='proportionAreaSearched',
maxDistance=100)
```

---

geometricRectanglePropSearchTable

*Create proportion of area searched table for a rectangular full plot*

---



---

 geometricRoadPadPropSearchTable

*Calculate the areas of intersection of a series of nested annuli with an idealized access road and turbine pad.*

---

### Description

Calculate area of annulus bisected by 2 parallel lines (e.g. a road of a road/pad plot).

### Usage

```
geometricRoadPadPropSearchTable(
  padRadius,
  roadWidth,
  maxSearchRadius,
  mastRadius,
  annulusWidth = 1,
  ...
)
```

### Arguments

padRadius	Integer, radius of turbine pad from the center of the turbine.
roadWidth	Integer, width of road leading to turbine pad.
maxSearchRadius	Integer, maximum search distance from the center of turbine.
mastRadius	Integer, radius of the turbine mast.
annulusWidth	Integer, width of annulus, default is 1.
...	Currently ignored.

### Details

Searches are conducted on the road and turbine pad around wind turbines for bird and bat fatalities. This function creates a data frame of proportion of area searched within each annulus ring on an idealized road and pad. The turbine is assumed to be centered on a perfectly circular turbine pad with radius padRadius, and a perfectly straight access road of width roadWidth is oriented from the center of the circle away from the turbine. (The resulting road and pad looks like rather like a lollipop.)

The mastRadius argument is to account for the area taken up by the turbine mast.

The arguments padRadius, roadWidth, mastRadius, and annulusWidth are all rounded to the nearest integer. The maxSearchDistance is rounded up (ceiling function) to an integer. If half units are needed, then convert to a smaller unit. See examples.

**Value**

Data frame of proportion of area searched for each annulus. distanceFromTurbine column represents the outer radius of each annulus.

**See Also**

geometricRectanglePropSearchTable circleBoxInt

**Examples**

```
pad <- 10 #meters, turbine pad radius
road <- 4 #meters, width of the road to the turbine pad
maxDistance <- 100 #meters, max distance
mast <- 2 #meters, turbine mast radius

## proportion are area searched at each annulus
propSearch <- geometricRoadPadPropSearchTable(padRadius = pad,
                                              roadWidth = road,
                                              maxSearchRadius = maxDistance,
                                              mastRadius = mast)

head(propSearch, 20)

## if half meter annulus rings are desired:
convert <- 100 # meters * 100 = centimeters

## units in centimeters
propSearchHalfMeter <- geometricRoadPadPropSearchTable(padRadius = pad * convert,
                                                       roadWidth = road*convert,
                                                       maxSearchRadius = maxDistance * convert,
                                                       mastRadius = mast * convert,
                                                       annulusWidth = 50) ##50cm = half a meter

head(propSearchHalfMeter, 30)

## convert back to meters
propSearchHalfMeter$distanceFromTurbine <- propSearchHalfMeter$distanceFromTurbine/convert
head(propSearchHalfMeter, 30)
```

---

getDistanceProbability

*Calculate probabilities within one unit increments.*

---

**Description**

Probabilities are calculated between specified increments for a given distribution and parameter values.

**Usage**

```
getDistanceProbability(
  q,
  distribution,
  param1,
  param2 = NA,
  tbound = c(-Inf, Inf),
  unitSize = 1,
  ...
)
```

**Arguments**

q	Vector of quantiles.
distribution	Character value; specifying the desired probability distribution, see <a href="#">ptrunc</a> .
param1	Numeric; value of the first parameter of the specified distribution.
param2	Numeric; default is NA. Value of the second parameter of the specified distribution, if applicable.
tbound	Numeric vector specifying the lower and upper truncation bounds. Default is <code>c(-Inf, Inf)</code> .
unitSize	Numeric; either of length one or equal to <code>length(q)</code> , specific the desired width for the probability calculation. Default is 1.
...	Currently ignored.

**Details**

This is a wrapper function that uses the [ptrunc](#) function. The basic calculation is `ptrunc(q, ...)`  
`-ptrunc(q-abs(unitSize), ...)`

**Value**

Vector of probabilities

**See Also**

[ptrunc](#)

**Examples**

```
## normal distribution
getDistanceProbability(q=8,distribution='norm',param1=10,param2=1)
pnorm(8,mean=10,sd=1)-pnorm(8-1,mean=10,sd=1)

## larger unitSize
getDistanceProbability(q=12,distribution='norm',param1=10,param2=1,unitSize=4)
pnorm(12,mean=10,sd=1)-pnorm(12-4,mean=10,sd=1)
```

getDistributionFunction

*Getting distribution functions*

---

### Description

Determines if the distribution functions are available. This is intended for internal use only.

### Usage

```
getDistributionFunction(type, dist, ...)
```

### Arguments

type	Character, typically either 'r', 'q', 'p', or 'd'.
dist	Character, typically something like 'norm', 'gamma', etc.
...	Currently ignored.

### Details

It is determined that `paste0(type,dist)` is a function and returns that function. The nature of the returned function is not verified.

### Value

Function, the first function in the search path that matches the name `paste0(type,dist)`.

### Examples

```
fun <- getDistributionFunction(type="q",dist="norm")
```

---

getDistributionSummary

*Summary statistics from the fitted distribution*

---

### Description

Summary statistics are calculated for the distribution with parameter estimates. Right now only the median is produced.

### Usage

```
getDistributionSummary(distribution, paramVec, truncBounds = NULL, ...)
```

**Arguments**

distribution	String indicating which distribution to use.
paramVec	Numeric vector for the parameters associated with distribution. Assumed to be in the same order as the function indicated by distribution.
truncBounds	Numeric, indicating bounds for the area correction calculation, see details. Default is NULL, and the bounds are set to $c(0, \text{Inf})$ .
...	Additional arguments to <code>integrate</code> .

**Details**

The `truncBounds` argument defaults to zero as a lower bound and infinity as the upper bound. If a single value is provided, it is assumed as the upper bound with zero as the lower bound. If two or more values are provided, the `max(truncBounds)` is the upper bound and `min(truncBounds)` is the lower bound.

**Value**

Data frame with the summary statistics.

**Examples**

```
getDistributionSummary('norm', c(40, 25), truncBounds=c(-Inf, Inf))
getDistributionSummary('norm', c(40, 25), truncBounds=NULL)
getDistributionSummary('norm', c(40, 25), truncBounds=c(0, 30))
```

---

```
getProportionAreaSearched
```

*Create proportion of area searched table from spatial data*

---

**Description**

Calculate proportion of area searched around wind turbine based on turbine location data and polygons of search area.

**Usage**

```
getProportionAreaSearched(
  turbinePoints,
  turbineName,
  turbinePlots,
  turbineMastRadius,
  maxDistance
)
```

**Arguments**

turbinePoints	Spatial points object with with data frame indicating turbine names
turbineName	Character, indicating the variable name for the turbine names in turbinePoints and plot names in turbinePlots
turbinePlots	Spatial polygon objects indicating the search area around the turbine points
turbineMastRadius	Integer, radius of the turbine mast
maxDistance	Integer, indicating how far from the turbine that searches occurred

**Details**

The `sf` package is used to calculate overlapping areas between the searched area `turbinePlots` and one unit annulus around the `turbinePoints`. The annuli increase out to a distance of `maxDistance`.

Caution, the function does some basic checks on the spatial objects but it is assumed that the points and polygons do not have any boundry, geometry, or other issues.

**Value**

Data frame of proportion of area searched for each annulus around each turbine point. `distanceFromTurbine` column represents outer radius of each annulus.

**Examples**

```
data(turbineSpatial)

propSearch <- getProportionAreaSearched(turbinePoints=turbineSpatial$turbinePoints,
turbineName='turbName', turbinePlots=turbineSpatial$turbinePlots,
turbineMastRadius=2,maxDistance=10)
```

---

getStartValue

*Calculate the start values to be passed to the optimizer.*

---

**Description**

Calculate start values for `weightedLikelihood` or `weightedDistribution`.

**Usage**

```
getStartValue(x, distribution, w = rep(1, length(x)), ...)
```



**Arguments**

<code>x</code>	Numeric vector of the data observations.
<code>distribution</code>	String indicating which distribution to use.
<code>w</code>	Numeric Vector of weights. This is assumed to be in the same order as <code>x</code> . Default is a vector of ones the same length as <code>x</code> .
<code>...</code>	Currently ignored.

**Details**

This function is intended for internal purposes only and is called by [weightedLikelihood](#) and [weighted-Distribution](#). The function calculates the weighted mean and weighted variance and performs a method of moments approach to obtain start values for the likelihood estimation.

**Value**

Vector of estimated parameters for the distribution.

**Examples**

```
x <- rnorm(100,10,5)

getStartValue(x, 'norm')
mean(x)
sd(x)
```

---

 hallingstad

*Data sets from Hallingstad et al. 2018*


---

**Description**

List containing two data frames from Hallingstad et al. 2018

**Usage**

```
data(hallingstad)
```

**Format**

The carcass element is a data frame with 26 rows and 3 variables:

**study** The name of the study for that wind farm.

**species** The species of the carcass found.

**distanceFromTurbine** The distance (meters) from the turbine of the location of the found carcass.

The weight element is a data frame with 2414 rows and 7 variables:

**study** The name of the study for that wind farm.

**distanceFromTurbine** The distance (meters) from the turbine for the outer radius of a one meter ring or annulus.

**detectionProbability** The probability of detection, incorporating search efficiency and carcass persistence, for a given study and distance from turbine.

**proportionAreaSearch** The proportion of area searched within the ring (annulus).

**scaledNumberTurbine** The number of turbines at a study divided by the total number of turbines for all studies considered, does not change with distance from turbine.

**scaledFatalityRate** The fatality rate at a study divided by the sum of the fatality rates across all studies considered, does not change with distance from turbine.

**finalWeight** The product of detectionProbability\*proportionAreaSearched\*scaledNumberTurbine\*scaledFatalityRate

## Details

A list of data frames, with two elements named carcass and weight.

## References

Hallingstad EC, Rabie PA, Telander AC, Roppe JA, Nagy LR (2018) Developing an efficient protocol for monitoring eagle fatalities at wind energy facilities. PLoS ONE 13(12): e0208700. <https://doi.org/10.1371/journal.pone.0208700>

---

hullMuirAreaCorrection

*Calculate an area correction based on the Hull and Muir (2010) maximum distance and a triangular distribution as proposed by Huso and Dalthorp (2014).*

---

## Description

Calculate the maximum fall distance from a turbine using the regression model from Hull and Muir (2010). Calculate the carcass fall probabilities between one-unit increments of a right triangle distribution as proposed by Huso and Dalthorp (2014). Use the probabilities and proportion of area searched to calculate an area correction value.

## Usage

```
hullMuirAreaCorrection(
  hubHeight,
  bladeRadius,
  lowerBound = 0,
  upperBound = Inf,
  proportionSearchDF,
  distanceCol,
```

```

    proportionCol,
    additionalCol = NULL,
    ...
)

```

### Arguments

hubHeight	Numeric, turbine hub height.
bladeRadius	Numeric, turbine blade radius.
lowerBound	Numeric, default is zero, see <a href="#">triangleProb</a> .
upperBound	Numeric, default is Inf, see <a href="#">triangleProb</a> .
proportionSearchDF	Data frame with at least two columns: distance from turbine and proportion of area searched at each distance.
distanceCol	Character string indicating the distance column in proportionSearchDF.
proportionCol	Character string indicating the proportion column in proportionSearchDF.
additionalCol	Character vector, default is NULL, indicating additional columns of how the area correction value should be calculated, see examples.
...	Currently ignored.

### Details

The maximum Hull and Muir distances are calculated using [hullMuirMaxDistance](#) and the carcass fall probabilities are calculated using [triangleProb](#). The probabilities are multiplied by the proportion of area searched from `proportionSearchDF` by distance. These products are summed across distances by size class and `additionalCol`.

The distances in the `distanceCol` will be rounded to the nearest integer for matching up with the probabilities. The distances, `hubHeight`, and `bladeRadius` are assumed to be in the same units.

### Value

Data frame of size class, `additionalCol` columns, and area correction

### References

Hull, C. L., & Muir, S. (2010). Search areas for monitoring bird and bat carcasses at wind farms using a Monte-Carlo model. *Australasian Journal of Environmental Management*, 17(2), 77-87.

Huso, M. & Dalthorp, D (2014). Accounting for Unsearched Areas in Estimating Wind Turbine-Caused Fatality. *The Journal of Wildlife Management*. 78. 10.1002/jwmg.663.

### See Also

[hullMuirMaxDistance](#) [triangleProb](#)

**Examples**

```
## proportion of area searched data
data(proportionAreaSearched)

hullMuirAreaCorrection(hubHeight = 87.5, bladeRadius = 62.5,
  proportionSearchDF = proportionAreaSearched,
  distanceCol = 'distanceFromTurbine',
  proportionCol = 'proportionAreaSearched',
  additionalCol = 'plotType')

## without additional columns but must separate the proportion of area searched
## data frame
hullMuirAreaCorrection(hubHeight = 87.5, bladeRadius = 62.5,
  proportionSearchDF = subset(proportionAreaSearched, plotType == 'RP'),
  distanceCol = 'distanceFromTurbine',
  proportionCol = 'proportionAreaSearched')

hullMuirAreaCorrection(hubHeight = 87.5, bladeRadius = 62.5,
  proportionSearchDF = subset(proportionAreaSearched, plotType == 'FULL'),
  distanceCol = 'distanceFromTurbine',
  proportionCol = 'proportionAreaSearched')
```

---

hullMuirMaxDistance     *Calculate the Hull and Muir (2010) maximum distance*

---

**Description**

Calculate the maximum fall distance from a turbine using the regression model from Hull and Muir (2010).

**Usage**

```
hullMuirMaxDistance(hubHeight, bladeRadius, ...)
```

**Arguments**

hubHeight	Numeric, turbine hub height.
bladeRadius	Numeric, turbine blade radius.
...	Currently ignored.

**Details**

Using the linear regression coefficients from Hull and Muir (2010), a maximum distance is calculated. This is done for three size classes (bats, small birds (SB), and large birds (LB)) separately.

It is assumed that hubHeight and bladeRadius have the same units.

Note: Hull and Muir (2010) used the range of 65 m < hubHeight < 94 m and 33 m < bladeRadius < 55 m. Anything outside of this range is extrapolation and should only be done with care.

**Value**

data frame of maximum distance by size class, hubHeight, and bladeRadius. Distance will be in the same units as were provided for hubHeight and bladeRadius

**References**

Hull, C. L., & Muir, S. (2010). Search areas for monitoring bird and bat carcasses at wind farms using a Monte-Carlo model. *Australasian Journal of Environmental Management*, 17(2), 77-87.

**Examples**

```
hubHeights <- rnorm(10, mean = 87.5, sd = 10)
bladeRadii <- rnorm(10, mean = 62.5, sd = 10)

hullMuirMaxDistance(hubHeight = hubHeights, bladeRadius = bladeRadii)
```

---

 LogLogistic

*Log-Logistic Distribution*


---

**Description**

The probability density function, cumulative density function, inverse cumulative density function, random generation for the log logistic distribution.

**Usage**

```
dlllog(x, shape = 1, scale = 1, log = FALSE, ...)
llogSummaryStats(shape, scale)
pllog(q, shape = 1, scale = 1, lower.tail = TRUE, log.p = FALSE, ...)
qllog(p, shape = 1, scale = 1, lower.tail = TRUE, log.p = FALSE, ...)
rllog(n, shape = 1, scale = 1, ...)
```

**Arguments**

x	Vector of quantiles.
shape	Shape parameter.
scale	Scale parameter.
log	Logical; if TRUE, log densities are returned.
...	Currently ignored.
q	Vector of quantiles.
lower.tail	Logical; if TRUE (default), probabilities are $P(X \leq x)$ otherwise, $P(X > x)$ .

log.p	Logical; if TRUE, probabilities p are given as log(p).
p	Vector of probabilities.
n	Number of observations. If length(n) > 1, the length is taken to be the number required.

### Details

If X is a random variable distributed according to a logistic distribution, then  $Y = \exp(X)$  has a log-logistic distribution.

The log-logistic distribution with parameters shape = a and scale = s has density

$$f(x) = \frac{\left(\frac{1}{a \cdot \exp(s)}\right) \left(\frac{x}{\exp(s)}\right)^{\frac{1}{a}-1}}{\left(1 + \left(\frac{x}{\exp(s)}\right)^{1/a}\right)^2}$$

for  $x \geq 0$ ,  $a > 1$ , and  $s > 0$ .

The median is  $\exp(s)$ , mean is

$$\frac{a\pi \cdot \exp(s)}{\sin(a \cdot \pi)}$$

for  $1/a > 1$ . The variance is

$$(\exp(s))^2 \left( \frac{2 \cdot \pi \cdot a}{\sin(2 \cdot \pi \cdot a)} - \frac{(a \cdot \pi)^2}{\sin^2(a \cdot \pi)} \right)$$

for  $1/a > 2$ . The mode is

$$\exp(s) \left( \frac{(1/a) - 1}{(1/a) + 1} \right)^a$$

for  $1/a > 1$  otherwise it is zero.

### Value

dllog returns vector of the densities.

pllog returns a vector of probabilities.

qllog returns a vector of quantiles.

rlllog returns a vector of random log-logistic variates.

### See Also

[dlogis](#) [plogis](#) [qlogis](#) [rlogis](#)

### Examples

```

y <- rlllog(5, shape=1, scale=1/3)
dllog(x=y, shape=1, scale=1/3)
dlogis(x=log(y), location=1/3, scale=1)/y

pllog(q=y, shape=1, scale=1/3)
qllog(p=seq(0, 1, by=.25), shape=1, scale=1/3)

```

---

print.windAC	<i>Print windAC object</i>
--------------	----------------------------

---

**Description**

Print windAC object.

**Usage**

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

**Arguments**

x	A windAC object.
...	Currently ignored.

**Details**

see [calcAC](#)

**Value**

Print for windAC object

---

proportionAreaSearched	<i>Proportion of area searched example data set</i>
------------------------	---

---

**Description**

An example data set of the proportion of area searched within 1 m annuli around turbines at a wind farm.

**Usage**

```
data(proportionAreaSearched)
```

**Format**

A data frame with 400 rows and 3 variables:

**plotType** Either FULL or RP, the type of plot searched at the turbine

**distanceFromTurbine** The outer radius distance from the turbine, for the one unit annulus.

**proportionAreaSearched** The proportion of area searched within the annulus at the turbine.

**Details**

Plots around turbines are searched for bird and bat carcasses. The area searched is summarized into the proportion of area searched with one unit annuli (or rings). The distance here corresponds to the outer radius of the annuli.

For example: plotType distanceFromTurbine portionAreaSearched RP 11 0.12441226

This row is corresponds to the annulus with outer radius of 11 and inner radius of 10. The proportion of area search on RP (road and pad) plot types is 0.12441226, or approximately 12 percent.

---

secondDerivative      *secondDerivative*

---

**Description**

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

**Usage**

```
secondDerivative(loc, FUN, ..., eps = 1e-07)
```

**Arguments**

loc	The location (a vector) where the second derivatives of FUN are desired.
FUN	An R function to compute second derivatives for. This must be a function of the form FUN <- function(x, ...) where x is the parameters of the function (e.g., location loc). FUN must return a single value (scalar), the height of the surface above x, i.e., FUN evaluated at x.
...	Additional arguments to FUN.
eps	Radius argument, see details. Default is 10e-7.

**Details**

This function uses the "5-point" numeric second derivative method advocated in numerous numerical recipe texts. During computation of the second derivative, FUN will be evaluated at locations within a hyper-ellipsoid with cardinal radius  $2 \cdot \text{loc} \cdot (\text{eps})^{0.25}$ .

A handy way to use this function is to call an optimization routine like `nlm` with FUN, then call this function with the optimized values (solution) and FUN. This will yield the hessian at the solution rather than the hessian at the previous step of the optimization.

**Value**

Matrix



**Examples**

```

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

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

func <- function(x){x[1]^2 + 5*x[2]^2} # should be rbind(c(2,0),c(0,10))
secondDerivative(c(1,1),func)
secondDerivative(c(4,9),func)

```

---

triangleProb	<i>Calculate probabilities from a triangle distribution based on Hull and Muir (2010) maximum distance as proposed by Huso and Dalthorp (2014).</i>
--------------	---

---

**Description**

Calculate the probabilities between one-unit increments of a right triangle distribution.

**Usage**

```
triangleProb(hubHeight, bladeRadius, lowerBound = 0, upperBound = Inf, ...)
```

**Arguments**

hubHeight	Numeric, turbine hub height.
bladeRadius	Numeric, turbine blade radius.
lowerBound	Numeric, default is zero, see Details.
upperBound	Numeric, default is Inf, see Details.
...	Currently ignored.

**Details**

A right triangle is constructed with the 90 degree corner at the origin in the first quadrant of the cartesian plane. The lowerBound will move the left edge of the triangle to the right. The upperBound will truncate the triangle distribution at that value.

The maximum horizontal distance is calculated using [hullMuirMaxDistance](#). This is typically not a whole number and the [ceiling](#) is used. The maximum vertical distance is such that the area under the hypotenuse edge of triangle integrates to one. This is done using the equation for the area of a triangle.

The two points that make up the hypotenuse are used to calculate the slope and intercept of the line. The area under the line in one-unit increments is calculated using

$$\int_{x-1}^x mZ + b dZ = m(x - .5) + b$$

where  $m$  is the slope,  $b$  is the intercept, and  $x$  is a distance. Integrating between  $x-1$  and  $x$  gives the probability between the one-unit increments.

All of this is done for three size classes (bats, small birds (SB), and large birds (LB)) separately. An additional size class (RAPTOR) is included and identical to the large bird result.

The `floor` function is applied to `lowerBound`.

It is assumed that `hubHeight` and `bladeRadius` have the same units.

### Value

List of two data frames: the first has distances in one-unit increments (the outer distance), the probabilities between the distances, and a column indicating size class; the second gives the maximum distance of each size class.

### References

Hull, C. L., & Muir, S. (2010). Search areas for monitoring bird and bat carcasses at wind farms using a Monte-Carlo model. *Australasian Journal of Environmental Management*, 17(2), 77-87.

Huso, M. & Dalthorp, D (2014). Accounting for Unsearched Areas in Estimating Wind Turbine-Caused Fatality. *The Journal of Wildlife Management*. 78. 10.1002/jwmg.663.

### See Also

[hullMuirMaxDistance](#)

### Examples

```
triResult <- triangleProb(hubHeight = 100, bladeRadius = 50, lowerBound = 0)
names(triResult) ## list names
triResult$maxDist ## max distance for each size class
head(triResult$triDistProb)
```

---

truncatedDistribution *Truncated Distributions*

---

### Description

Truncated probability density function, truncated cumulative density function, inverse truncated cumulative density function, and random variates from a truncated distribution.

**Usage**

```
dtrunc(x, distribution, tbound = c(-Inf, Inf), ..., log = FALSE)
```

```
ptrunc(
  q,
  distribution,
  tbound = c(-Inf, Inf),
  ...,
  lower.tail = TRUE,
  log.p = NULL
)
```

```
qtrunc(
  p,
  distribution,
  tbound = c(-Inf, Inf),
  ...,
  lower.tail = TRUE,
  log.p = NULL
)
```

```
rtrunc(n, distribution, tbound = c(-Inf, Inf), ...)
```

**Arguments**

x	Vector of quantiles.
distribution	Character value specifying the desired probability distribution.
tbound	Numeric vector specifying the lower and upper truncation bounds. Default is <code>c(-Inf, Inf)</code> .
...	Additional arguments passed to the non-truncated distribution functions.
log	Logical; if TRUE, log densities are returned.
q	Vector of quantiles.
lower.tail	Logical; if TRUE (default), probabilities are $P(X \leq x)$ otherwise, $P(X > x)$ .
log.p	Currently ignored.
p	Vector of probabilities.
n	A positive integer specifying the desired number of random variates.

**Details**

The non truncated distribution functions are assumed to be available. For example if the normal distribution is desired then used `distribution='norm'`, the functions then look for 'qnorm', 'pnorm', etc.

The `max(tbound)` and `min(tbound)` are considered the upper and lower truncation bounds, respectively.

The random variates are produced using the direct method (see Casella and Berger 2002).

**Value**

dtrunc returns a vector of densities.

ptrunc returns a vector of probabilities.

qtrunc returns a vector of quantiles.

rtrunc returns a vector of random variates.

**References**

G. Casella and R. L. Berger. Statistical inference. Vol. 2. Duxbury Pacific Grove, CA, 2002.

**Examples**

```
## dtrunc
# not truncated
dnorm(5,mean=5)
# truncated
dtrunc(x=5,distribution='norm',tbound=c(4,5.5),mean=5)

## ptrunc
#not truncated
pgamma(2,shape=3,rate=2)
# truncated
ptrunc(2, distribution = 'gamma', tbound=c(1,5),shape=3,rate=2)

## upper tail
# not truncated
pgamma(2,shape=3,rate=2,lower.tail=FALSE)
# truncated
ptrunc(2,distribution='gamma',tbound=c(1,5),shape=3,rate=2,lower.tail=FALSE)

## qtrunc
#not truncated
qnorm(p=.975)
# truncated
qtrunc(p=.975,distribution='norm',tbound=c(0,1))

## upper tail
# not truncated
qnorm(p=.975,lower.tail=FALSE)
# truncated
qtrunc(p=.975,distribution='norm',tbound=c(0,1),lower.tail=FALSE)

## rtrunc
rtrunc(n=5, distribution = 'gamma', tbound=c(2,5),shape=3,rate=2)
```

---

turbineSpatial	<i>Data sets of spatial area searched at a wind farm</i>
----------------	--

---

**Description**

List containing two spatial data frames

**Usage**

```
data(turbineSpatial)
```

**Format**

An object of class `list` of length 2.

**Details**

A list of data frames, with two elements named `turbinePlots` and `turbinePoints`. These are fictitious data for example purposes only. The `turbinePlots` element is a spatial polygon data frame with the polygons being the area searched round the turbines. The `turbinePoints` element is a spatial points data frame with the turbine locations.

---

weightFun	<i>weight function</i>
-----------	------------------------

---

**Description**

Generic weight function for use with [estWD](#).

**Usage**

```
weightFun(  
  x,  
  propTable,  
  type,  
  typeCol,  
  distanceCol,  
  propCol,  
  xFun = ceiling,  
  maxDistance = NULL,  
  ...  
)
```

**Arguments**

x	Numeric vector.
propTable	Data frame contain the proportion of area searched by distance and plot type.
type	Character, indicating which plot type to subset propTable.
typeCol	Character, column name of the plot type in propTable.
distanceCol	Character, column name of the distance in propTable.
propCol	Character, column name of the proportion of area searched in propTable.
xFun	Function, default is <a href="#">ceiling</a> , see details.
maxDistance	Numeric, default is NULL. If a value is given then propTable is subsetted to where propTable[,distanceCol] <= maxDistance.
...	Additional arguments passed to xFun.

**Details**

The [weightedDistribution](#) function requires the weights be described using a function. This allows integration to happen.

Typically propTable has integer values for the distances, but the function needs to take in any numeric values, the xFun function is how any numeric value can be matched up to the values in propTable. If the distances in propTable correspond to the outer radius of the annuli, for calculating proportion of area searched, then the [ceiling](#) is appropriate. If the distances in propTable correspond to the inner radius of the annuli then the [floor](#) might be more appropriate.

**Value**

Numeric vector of weights with length equal to length(x), and with a 1:1 relationship to the values in x.

**See Also**

[estWD](#)

**Examples**

```
data(proportionAreaSearched)

d <- c(-300.23,14.3,16,75)

## RP proportion of area searched
weightFun(x=d,propTable=proportionAreaSearched,type='RP',typeCol='plotType',
distanceCol='distanceFromTurbine',propCol='proportionAreaSearched')
#[1] 0.00000000 0.08896480 0.08308577 0.01709869

## FULL plot proportion of area searched
weightFun(x=d,propTable=proportionAreaSearched,type='FULL',typeCol='plotType',
distanceCol='distanceFromTurbine',propCol='proportionAreaSearched')
# [1] 0 1 1 1
```

```

### with a max distance restriction
## RP proportion of area searched
weightFun(x=d,propTable=proportionAreaSearched,type='RP',typeCol='plotType',
distanceCol='distanceFromTurbine',propCol='proportionAreaSearched',maxDistance=40)
# [1] 0.00000000 0.08896480 0.08308577 0.00000000

## FULL plot proportion of area searched
weightFun(x=d,propTable=proportionAreaSearched,type='FULL',typeCol='plotType',
distanceCol='distanceFromTurbine',propCol='proportionAreaSearched',maxDistance=40)
# [1] 0 1 1 0

```

---

windAC

*A package for calculating area correction values for fatality estimation at wind farms.*

---

## Description

Post-construction fatality monitoring studies at wind facilities are based on data from searches for bird and bat carcasses in plots beneath turbines. Bird and bat carcasses can fall outside of the search plot. Bird and bat carcasses from wind turbines often fall outside of the searched area. To compensate, area correction (AC) estimations are calculated to estimate the percentage of fatalities that fall within the searched area versus those that fall outside of it. This package provides two likelihood based methods and one physics based method (Hull and Muir (2010), Huso and Dalthorp (2014)) to estimate the carcass fall distribution. There are also functions for calculating the proportion of area searched within one unit annuli, log logistic distribution functions, and truncated distribution functions.

The two likelihood methods are the truncated weighted likelihood ([estTWL](#)) and the weighted distribution ([estWD](#)). Both use carcass distances from the turbine, accounting for unequal detection by distance, to estimate the distance distribution. Alternatively, a right triangle distribution can be used for the carcass density distribution with the max distance estimated ([hullMuirMaxDistance](#) from the regression from Hull and Muir (2010) as proposed by Huso and Dalthorp (2014).

The area correction value is calculated from the combination of the carcass distance density and the proportion of area searched at each distance. The function [getProportionAreaSearched](#) uses the [sf](#) package to do this from turbine points spatial data and search area polygons. The functions [geometricRectanglePropSearchTable](#) and [geometricRoadPadPropSearchTable](#) also calculate proportion of area searched but assuming perfect geometric shapes, meaning no spatial data is required.

Search areas are often irregular. [proportionAreaSearched](#) summarizes the area searched into the proportion of area searched with one unit annuli (or ring).

Two sets of distribution functions are available. Log logistic distribution functions (see [dllog](#)). These are a transformation of the logistic distribution and use the base R functions (see [dlogis](#)). The second is truncation functions (see [dtrunc](#)), that provide truncation for R function distributions.

Example data sets:

- **carcassDistance**: Example data set of carcass distances found during searches under wind turbines.
- **hallingstad**: List containing two data frames from Hallingstad et al. 2018. Data frames list carcass and detection probability data for a few wind farm studies.
- **proportionAreaSearched**: An example data set of the proportion of area searched around turbines at a wind farm.

## References

- Hallingstad EC, Rabie PA, Telander AC, Roppe JA, Nagy LR (2018) Developing an efficient protocol for monitoring eagle fatalities at wind energy facilities. PLoS ONE 13(12): e0208700. <https://doi.org/10.1371/journal.pone.0208700>
- Huso, M. & Dalthorp, D (2014). Accounting for Unsearched Areas in Estimating Wind Turbine-Caused Fatality. The Journal of Wildlife Management. 78. 10.1002/jwmg.663.
- Hull, C. L., & Muir, S. (2010). Search areas for monitoring bird and bat carcasses at wind farms using a Monte-Carlo model. Australasian Journal of Environmental Management, 17(2), 77-87.



# Index

- \* **datasets**
  - carcassDistance, 4
  - hallingstad, 17
  - proportionAreaSearched, 23
  - turbineSpatial, 29
- \* **package**
  - windAC, 31
- calcAC, 2, 7, 9, 23
- carcassDistance, 4, 32
- ceiling, 25, 30
- circleBoxInt, 5
- dllog, 31
- dllog (LogLogistic), 21
- dlogis, 22, 31
- dtrunc, 31
- dtrunc (truncatedDistribution), 26
- estTWL, 6, 7, 31
- estWD, 8, 29–31
- floor, 26, 30
- geometricRectanglePropSearchTable, 9, 31
- geometricRoadPadPropSearchTable, 11, 31
- getDistanceProbability, 3, 4, 12
- getDistributionFunction, 14
- getDistributionSummary, 14
- getProportionAreaSearched, 15, 31
- getStartValue, 16
- hallingstad, 17, 32
- hullMuirAreaCorrection, 18
- hullMuirMaxDistance, 19, 20, 25, 26, 31
- integrate, 15
- llogSummaryStats (LogLogistic), 21
- LogLogistic, 21
- optim, 7, 8
- pllog (LogLogistic), 21
- plogis, 22
- print.windAC, 23
- proportionAreaSearched, 23, 31, 32
- ptrunc, 13
- ptrunc (truncatedDistribution), 26
- qllog (LogLogistic), 21
- qlogis, 22
- qtrunc (truncatedDistribution), 26
- rllog (LogLogistic), 21
- rlogis, 22
- rmvnorm, 3
- rtrunc (truncatedDistribution), 26
- secondDerivative, 24
- sf, 16, 31
- triangleProb, 19, 25
- truncatedDistribution, 26
- turbineSpatial, 29
- weightedDistribution, 4, 16, 17, 30
- weightedDistribution (estWD), 8
- weightedLikelihood, 4, 16, 17
- weightedLikelihood (estTWL), 6
- weightFun, 8, 9, 29
- windAC, 31